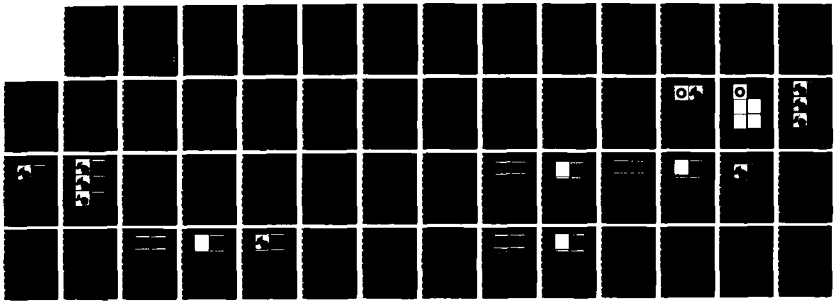
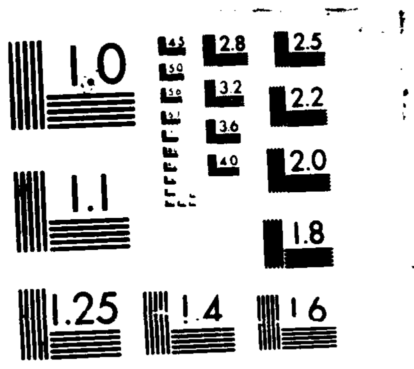
AD-A179 876 ADVANCED LIKELIHOOD GENERATORS FOR BOUNDARY DETECTION 1/1
(U) ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE D SHER
JAN 87 TR-197 DACA76-85-C-0001

UNCLASSIFIED F/G 12/9 NL




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

12

Advanced Likelihood Generators for Boundary Detection

David Sher
Computer Science Department
The University of Rochester
Rochester, New York 14627

January 1987
TR 197

rochester

Department of Computer Science
University of Rochester
Rochester, New York 14627

Advanced Likelihood Generators for Boundary Detection

David Sher
Computer Science Department
The University of Rochester
Rochester, New York 14627

January 1987
TR 197

In [Sher85] I discussed the advantages of feature detectors that return likelihoods. In [Sher86] I demonstrated some preliminary results with a boundary point detector that returns likelihoods. Now I have developed boundary point detectors that have these properties:

- Return probabilities
- Can be combined robustly
- Potential sub-pixel precision
- Work with correlated noise.
- Can handle multiple gray-levels (tested with 255)

These detectors were applied both to aerial images and to test images whose boundaries are known. They are compared to established edge detectors.

This work would have been difficult if not impossible without the active support and encouragement of Chris Brown, my advisor, and Jerry Feldman. This work was supported by the Defense Advanced Research Projects Agency U. S. Army Engineering Topographic Labs under grant number DACA76-85-C-0001. Also this work was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss Air Force Base, New York 13441-5700, and the Air Force Office of Scientific Research, Bolling AFB, DC 20332, under Contract No. F30602-85-C-0008. This contract supports the Northeast Artificial Intelligence Consortium (NAIC).

A179876

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR 197	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Advanced Likelihood Generators for Boundary Detection		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) David Sher		8. CONTRACT OR GRANT NUMBER(s) DACA76-85-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department The University of Rochester Rochester, New York 14627		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA/1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE January 1987 13. NUMBER OF PAGES 52
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

edge detection, template, likelihood, Bayesian reasoning.

— [illegible] were

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In(Sher85) I discussed the advantages of feature detectors that return likelihoods. In(Sher86) I demonstrated some preliminary results with a boundary point detector that returns likelihoods. Now I have developed boundary point detectors that have these properties:

*Return probabilities

*Can be combined robustly

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (Continued)

*Potential sub-pixel precision

*Work with correlated noise.

*Can handle multiple gray-levels (tested with 255).

These detectors were applied both to aerial images and to test images whose boundaries are known. They are compared to established edge detectors.

[illegible]

1. Introduction

Currently a great variety of tools are available for low-level vision tasks such as image reconstruction and edge detection. It is time to devote attention to managing tools rather than creating new ones.

Most of the tools for low level vision are algorithms for intermediate steps towards achieving a goal. Here, we consider boundary point detection algorithms in these terms. These are algorithms that try to determine if a boundary passes through a pixel (usually given a window on the image). This task is similar to edge detection. Boundary point detection algorithms do not exist to display outlines pleasing to the human eyes. Their output is meant to be input to a higher level routine such as a shape recognition program or a surface reconstruction program.

Some desiderata for boundary point detection tools are:

(1) The output of a boundary point detector should be useful to as many higher level routines as possible. If every higher level routine required a different boundary point detector then our technology has not lived up to this desiderata.

(2) Boundary point detectors should accept as input the full range of data available. If a boundary point detector only worked for binary data when gray scale data was available, it has not lived up to this desiderata.

(3) Boundary point detectors should do work proportional to the size of the image.

(4) Boundary point detectors should do work proportional to the precision of the output. Thus if subpixel precision is required, it should be made available with work proportional to the required accuracy of boundaries reports.

(5) Boundary point detectors should be parameterized to features of the data. For example, if the distribution of reflectances in the scene is known (the expected image histogram) then a detector should be constructed that uses this information. Another example is if structure in the noise is known (such as correlation) we should be able to take this structure into account.

In this paper I describe an algorithm that fits these desiderata. It is a more advanced version of the algorithm described in [Sher86]. Also the results of tests run on this algorithm are reported here and comparisons with established algorithms such as the Sobel, Kirsch and variants thereupon. Tests will be done soon on more sophisticated operators.

2. Definition of Boundary

Before talking about boundary point detector it is a good idea to define exactly what a boundary is. Vision problems involve imaging a scene. This scene could be an aerial view or a picture of machine parts or an outdoors scene. This scene is filled with *objects*. In an aerial photograph some of these objects are buildings, trees, roads and cars. Each of these objects is projected into the observed image. Thus each object has an *image* that is a subset of the observed image. Where the observed image of one object meets an observed image of another object there is a *boundary*. such boundaries are sometimes referred to as occlusion boundaries.

3. Returning Probabilities

The algorithm I describe fulfills the first desideratum by returning probabilities that a boundary is near a point. Here I justify returning probabilities and show how I can fulfill the first desideratum using them.

No tool for boundary point detection or any other low-level vision task ever does its task without a significant probability of being wrong. Thus the error characteristics of a low-level vision algorithm need to be considered. Intermediate-level vision algorithms differ in sensitivities to different kinds of errors.

Consider boundary point detection. Regularization algorithms (interpolation algorithms) [Boult6?] suffer more from missing a boundary point than from having extra ones. When a boundary point is missed regularization algorithms try to smooth over the boundary with disastrous results. Hough transform techniques often work effectively when the set of boundary points detected is sparse because the work a hough transform technique does is proportional to the size of that set.

Another reason that Hough transform techniques do well with sparse data is that they are mode based. Thus Hough transform techniques are robust when feature points are left out and when there are outlying data points. The robustness of the Hough transform is described in detail in [Brown82].

It is good software management to use the same boundary point detector to generate input for all high level vision tasks that require boundary point detection rather than building a special detector for each high level routine. If a boundary point detector returns a true/false decision for each point then its output does not suit both regularization techniques and hough transform techniques. Take for example the one dimensional intensity slice shown in Figure 1.

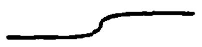


Figure 1: Slice through an image with an ambiguous boundary

For use as a first stage before regularization it is preferable that such ambiguous slices be considered boundaries because the cost of missing a boundary is high (compared to the cost of missing an edge). For use with hough transform line detection figure 1 is not a good boundary because the cost of an extra edge is high.

The traditional solution to the dilemma of satisfying differing requirements among intermediate level routines has been to supply numbers such as *edge strengths* rather than true/false decisions. These strengths describe how likely the low-level vision algorithm considers the event such as the existence of a boundary. The example in figure 1 has the detector return a low edge strength. Figure 2 shows a 0 edge strength.

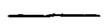


Figure 2: Slice through an image with an edge strength of 0

Figure 3 shows a high edge strength.

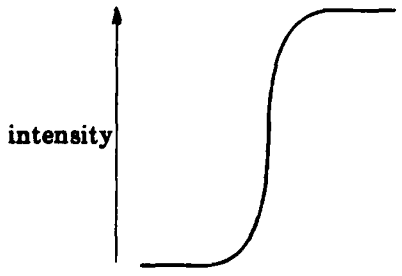


Figure 3: Slice through an image with a high edge strength

If an edge detector that returns strengths (acting as a boundary point detector) is used as input for various intermediate level applications, then each application usually uses a threshold to determine what is and isn't a boundary. If an edge strength is higher than the threshold a boundary is reported. The threshold is determined by running the boundary point detector and the intermediate level algorithm with different thresholds and finding the threshold that results in the best results according to some error norm or human observation[1]. If the boundary point detector is changed new thresholds need to be found.

If boundary point detectors were standardized so that the correspondence between strength and the probability of the boundary were consistent between all boundary point detectors then the threshold need only be calculated once for each intermediate level application. Then the thresholds for intermediate level applications could be calculated from theoretical principles. When the strengths have clear semantics the entire process of threshold determination would be fully understood.

A consistent and well defined output for boundary point detectors is the probability of a boundary. If all boundary point detectors output the probability of a boundary then the boundary point detector could be improved without changing the rest of the system. If the error sensitivities of the intermediate level routines are known the thresholds can be determined by a simple application of decision theory.

4. Likelihoods

Most models for boundary point detection describe how configurations of objects in the real world generate observed data. Such models do not explicitly state how to derive the configuration of the real world from the sensor data. This behavior of models results in graphics problems being considerably easier than vision problems. Thus we have programs that can generate realistic images that no program can analyze.

Given the assumption "There is a boundary between pixels p_1 and p_2" we can determine a probability distribution over possible observed images. Let $b(1,2)$ be the statement that there is a boundary between pixels p_1 and p_2. Let $S(b(1,2))$ be the set of region maps such that $b(1,2)$ is true (and generally I use the notation that $S(\textit{statement})$ is the set of region maps where *statement* is true). Let M be the model for the boundary detection task. Then the probability of O (the observed image) given $b(1,2)$ under M is calculated by equation 1.

[1]Edge relaxation algorithms often adjust the edge strengths. For the purposes of this paper consider such an algorithm as a part of the detector, the output of this detector is the strengths output by the relaxation algorithm.

$$P(O|b(1,2)\&M)=\frac{\sum_{i\in S(b(1,2))} P(O|i\&M)P(i|M)}{P(b(1,2)|M)} \tag{1}$$

If O is the observed data then the probability calculated in equation 1, $P(O|b(1,2)\&M)$, is called here (inspired by the statistical literature) the *likelihood* of $b(1,2)$ given observed data O under M. Generally the models we use in computer vision make the calculation of likelihoods simple for the features we want to extract. However the desired output of a feature detector is the conditional probability of the feature. Thus in boundary point detection I can derive $P(O|b(1,2)\&M)$ and I can derive $P(O|\sim b(1,2)\&M)$ ($\sim x$ means "not x" in this paper) but I want to derive $P(b(1,2)|O\&M)$.

A theorem of probability theory, *Bayes' law*, shows how to derive conditional probabilities for features from likelihoods and prior probabilities. Bayes' law is shown in equation 2.

$$P(f|O\&M)=\frac{P(O|f\&M)P(f|M)}{P(O|f\&M)P(f|M)+P(O|\sim f\&M)P(\sim f|M)} \tag{2}$$

In equation 2 f is the feature for which we have likelihoods. M is the model we are using. $P(O|f\&M)$ is the likelihood of f under M and $P(f|M)$ is the probability under M of f (the prior probability).

For features that can take on several mutually exclusive labels such as surface orientation a more complex form of Bayes' law shown in equation 3 yields conditional probabilities from likelihoods and priors.

$$P(l|O\&M)=\frac{P(O|l\&M)P(l|M)}{\sum_{l'\in L(f)} P(O|l'\&M)P(l'|M)} \tag{3}$$

l is a label for feature f and $L(f)$ is the set of all possible labels for feature f.

Likelihoods are important because they are a useful intermediate term on the way to deriving a posterior probability. An upcoming technical report describes formulas for evidence combination based on likelihoods [Sher87]. To apply that theory of evidence combination one needs to compute the likelihoods explicitly. Thus in the succeeding sections I calculate the likelihoods even up to factors that are equal in all the likelihoods (hence are divided out by equation 3).

Another important use for explicit likelihoods is for use in Markov random fields. Markov random fields describe complex priors that can capture important information. Markov random fields were applied to vision problems in [Geman84]. Likelihoods can be used with a Markov random field algorithm to derive estimates of boundary positions [Marroquin85] [Chou87].

5. Simplifications

To make the problem of computing the likelihoods for events computationally tractable, I simplify my problem in certain ways.

The first simplification is calculating the probability of a boundary at a point rather than trying to compute a probability distribution over boundary maps for the entire scene. Even though a distribution over complete maps would be more general there are too many possible boundary maps to manage realistically. Under certain circumstances all that is needed is to compute the probability of a boundary near each pixel [Marroquin85].

5.1. Window Based Boundary Detectors

Equation 1 implies an algorithm for determining likelihoods of boundaries. It shows that iterating through all the configurations that cause a boundary at a point is a way to find the likelihood of a boundary. A *region map* is a description of where the images of the objects are. Each configuration of objects in a scene implies a region map. However, the set of region maps that contain a boundary between two pixels is too large to iterate through (for a 512 by 512 observed image it is $>>10^{25,000}$). The cardinality of the set of region maps is in general exponential in the size of the image.

An obvious solution to the problem is to reduce the number of pixels. Generally, the further one gets from a boundary the less relevant the data in the image is to that boundary. Thus it is common to use a relatively small window about the proposed boundary for finding the boundary (see figure 4). There are many fewer region maps over a 4 by 4 window than over a 512 by 512 image.

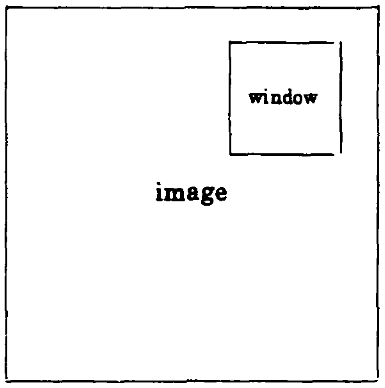


Figure 4: Small Window on Large Image

Thus by looking only at a window I simplify the problem of boundary point detection considerably. Inevitably, I lose some accuracy in the computation of probabilities from limiting the data to a window. However the simplification of the algorithm compensates for this loss. Every attempt at edge detection has used this principle [Hueckel71] [Canny83] with possibly a further stage of linking or relaxation.

5.2. Constraining Region Maps

Applying a boundary point detector to an H (height) by D (depth) observed image involves computing the probability of HD boundaries. If $H=D=512$, then $HD=262144$. A boundary point detection algorithm computes probabilities for all these potential boundaries. The cost of using algorithm that computes the probability of a boundary at a point is multiplied by HD. The cost may be reduced by sharing some of the work between iterations. Still much of the work can not be shared. Saving time by sharing work between iterations is discussed in later sections.

Ignoring saving by sharing between iterations, any saving in the algorithm that computes the probability of a boundary at one point is multiplied manyfold (for $H=D=512$ 262144fold). Algorithms for computing the probability of a boundary at a point require work proportional to the number of region maps. Reducing the number of region maps that need to be considered proportionately reduces the work required by the algorithms for boundary point detection.

The maximal amount a region map can change the likelihood of a feature label is the probability of that region map divided by the prior probability of a feature label corresponding to that region map (shown by a cursory examination of equation 1, repeated below, with i being a region map).

$$P(O|b(1,2)\&M) = \frac{\sum_{i \in S(b(1,2))} P(O|i\&M)P(i|M)}{P(b(1,2)|M)} \qquad (1)$$

Bayes' law (equation 3) can be broken into two steps. First the likelihoods are multiplied by the priors. Consider the likelihood of a feature label (say *boundary*) multiplied by the prior probability. Call such a term the *conjunctive probability* of l since it is $P(O\&f=l\&M)$. The conjunctive probability of l can be changed by deleting a region map with $f=l$ at most the probability of that region map. The probability $P(f=l|O\&M)$ is derived from the conjunctive probabilities by dividing $P(f=l\&O\&M)$ by the sum of the conjunctive probabilities $\sum_r P(f=r\&O\&M)$. Thus if the probability of a region map is small compared to $P(O\&M) = \sum_r P(f=r\&O\&M)$ deleting the likelihood corresponding to it has a small effect on the resulting distribution. Thus one can with some safety ignore region maps whose prior probability is small enough.

For standard edge detection algorithms a common restriction on region maps is to assume that there are at most two object images participating in the window thus there can only be two regions [Hueckel71]. Step edge based models implicitly make this assumption [Canny83] [Nalwa84]. Windows with more than two object images in them are assumed to occur infrequently I call the assumption that there are at most two object images participating in a window the *two object assumption*.

Another simplification places a limitation on the curvature of the boundaries observed in the image. Limiting this parameter limits the set of region maps in a mathematically convenient way. If the curvature is limited enough the boundaries can be considered to be straight lines within windows. Thus the windows on the observed image can be modeled assuming there is no boundary or a single linear boundary across them. A similar model (it allowed two linear boundaries in a window) was used in [Hueckel71]. I call the assumption that there are no high curvature boundaries the *low curvature assumption*.

5.3. Numerical Approximations

Another effect that makes the probabilities calculated by my algorithms inaccurate is that a real number can only be specified to a limited accuracy on the computer. Thus there is a limit to the accuracy that calculations can be performed to in the computer. In section 6 I ignore the error introduced from inexact floating point computations. In my implementation (section 8) I used double precision arithmetic throughout in an attempt to reduce this error.

Another source of errors is my simplifying the mathematics to make the algorithm simpler. One such approximation I make is to use the density of a normal distribution as a probability. In the equations derived in section 6 I use this approximation in the probability derived from a multinormal Gaussian. This approximation simplified the mathematics for deriving the detector. Such an approximation is standard.

6. Building a Boundary Detector

In section 4 I discuss why determining likelihoods is a useful first step in feature detection. In section 5 I describe the approximations and simplifications necessary to make the problem of boundary point detection computationally tractable. Now all that is left is to develop the algorithm. The first step in deriving a boundary point detection algorithm is to derive the likelihood that a window is filled with a single object given the observed data (section 6.1). Then likelihoods for windows with multiple objects are derived (section 6.3). In this section I assume that the model has Gaussian mean 0 noise with a known standard deviation added to an ideal image.

6.1. Likelihoods for a Single Object

The problem is to find the likelihood of a single object filling a window in the image. The expected intensities of the pixels in the window are proportional to the reflectance of O. Let T_0 be the expected value of the pixels in the window when O has reflectance 1, $r(O)=1$. T_0 is often referred to as a *template* in the image understanding literature.

Template matching can be best shown on a two dimensional medium (with weak graphical capacities) when the window is 1 dimensional. Thus I use a 1 by 8 window for examples. A typical template for a single object in a 1 by 8 window is shown in figure 5.

100	100	100	100	100	100	100	100

Figure 5: Template for a single object

This template is boring because I assume that there is little variance in intensity in the image of the interior of an object. The observed image of the object has a normal iid added to each pixel. Thus the observed image (when the standard deviation is 8) can look like figure 6.

94	104	100	94	92	105	101	103

Figure 6: Noised Template for a single object

The probability that the noised window results from the template is a function of the vector difference between the window and the template. For the example in figure 6 given the template for a single object in figure 5, the probability is calculated by summing the squared differences between the template and the observed data (187) and then applying the normal distribution function (for 8 independent normally distributed samples mean 0 standard deviation 8 rounded to the nearest integer) to get 8.873e-12.

In later sections when I use windows or templates in equations the windows or templates are flattened into vectors. Thus a two dimensional window is transformed into a vector (figure 7).

1	2	3
4	5	6
7	8	9

Window 1

1	2	3	4	5	6	7	8	9

The Vector from Window 1.
Figure 7: Flattening a window into a vector

All the windows and templates are assumed to be flattened thus. When a template is subtracted from a window, vector subtraction is happening. When a template or window is being multiplied by a matrix, vector matrix multiplication is happening. In particular, WW^T is the sum of the squares of the elements of W while WT^T is the sum of the products of the corresponding elements of W and T.

Let aT_0 be a times the intensities in the template T_0. The template for an object with reflectance $r(O)$ is $r(O)T_0$. The probability of observing a window W when the scene is of O is determined by the Gaussian additive factor. Equation 4 is the formula for the probability. (Let V be the variance of the noise, σ^2 in the rest of this paper) Let n be the size of the window and κ be the constant $1/(2\pi V)^{n/2}$.

$$P(W|r(O)=r\&M)=\kappa\exp(-(W-rT_0)(W-rT_0)^T/2V) \tag{4}$$

Since the reflectance of the object in the scene is not known but the distribution of reflectances is known one must integrate this formula over possible reflectances. Thus the probability of a window given it is of a single object of known position and shape is in equation 5.

$$P(W|O\&M) = \int\kappa\exp(-(W-rT_0)(W-rT_0)^T/2V)dD_r(r) \tag{5}$$

The term $(W-rT_0)(W-rT_0)^T$ can be rearranged to $WW^T-2rT_0W^T+r^2T_0T_0^T$. WW^T is the sum of the squares of the pixels in the window. Let us refer to it as $\mathbf{W}^2$ for shorthand. T_0W^T is the correlation between T_0 and W. Let us refer to it as C for shorthand. $T_0T_0^T$ is the sum of the squares of the elements of the template. Let us refer to it as T^2. Using a rearrangement and these shorthands equation 5 can be restated as equation 6.

$$P(W|O\&M) = \exp(-\mathbf{W}^2/2V)\kappa\int\exp((2rC-r^2T^2)/2V)dD_r(r) \tag{6}$$

Equation 6 is the product of two parts, equation 7 :

$$\mathbf{F}_1(\mathbf{W}^2) = \exp(-W^2/2V)\kappa \tag{7}$$

and equation 8 :

$$F_2(\mathbf{C}) = \int\exp((2rC-r^2T^2)/2V)dD_r(r) \tag{8}$$

F_1 and F_2 have one parameter that depends on the window (T^2 is unchanged over all windows). They can be implemented by table lookup without excessive use of memory or much computation.

Hence the algorithm to calculate the likelihood of a window given it is of a single object of known position and shape (but unknown reflectance) is described by figure 8 below.

Figure 8: Algorithm to Calculate Likelihood of Known Object for a Single Window

			Multiplies	Adds
$\mathbf{W}^2$	:=	WW^T	w	$w-1$
C	:=	WT_0^T	w	$w-1$
Output		$F_1(W^2)*F_2(C)$	1	0

w is the number of pixels in W and also the number of pixels T_0. Figure 8 is a $4w-1$ operations algorithm.

6.2. Reducing the Configurations Considered

In section 4 I describe how to derive the likelihood of a boundary at a point from the likelihoods of configurations of a the scene given a boundary at a point (equation 1). Here I justify using a small number of configurations of the scene. Having reduced the set of configurations I can derive an efficient algorithm for generating likelihoods given multiple objects. In section 5.1 I made the assumption that only a small window on the image need be looked at. In section 5.2 I justified ignoring region maps with low probability.

There is some number of objects, N_O, such that the probability N_O or more objects having an image in a single window has a probability much smaller than the probability of all but a small probability subset of the region maps. Hence I need only consider configurations of $N_O - 1$ or less objects in a window. The two object assumption of section 5.2 is equivalent to saying $N_O = 3$.

There are still a large set of configurations of less than N_O objects. Consider the ideal image given such a configuration and some coloring of objects. Consider a window on this ideal image, W_I. There is a set of configurations with the same coloring such that the resulting window on the ideal window from such a configuration W_J such that:

$$(W_I - W_J)(W_I - W_J)^T < \varepsilon \tag{9}$$

The likelihood that the observed image results from any of the configurations that fit the criterion of equation 9 and coloring is close to the likelihood computed from W_I because if W is the observed window the likelihood is a function of the norm of $W - W_J$. Thus for efficiency sake we can consider the likelihood of each configuration and coloring that fit equation 9 the same as that of the configuration and coloring of the template that generates W_I. Hence we can only consider a small set of configurations of objects. With a similar argument one can prove that only a subset of the possible colorings need be considered. Hence I can justify using an argument of this sort using a small set of templates and objects. How much inaccuracy results from these simplifications can be analyzed mathematically.

A step edge model can be derived by assuming that objects' images have a uniform intensity and the two object assumption. Thus such simplifications underlies work like that of [Hueckel73] [Canny83]. More sophisticated assumptions about the intensities of objects images result in more complex models [Binford81] that still can be reduced to a reasonable number of configurations reflectances with a corresponding loss of accuracy in the resulting probabilities.

6.3. Algorithm for Likelihoods of Multiple Objects

Here I derive an algorithm for finding the likelihood that a scene that contains several objects given a window.

The statement that the configuration of objects is near the specified configuration is called **C**. **C** has N_O objects O_n with reflectances r_n. Associated with **C** are N_O templates T_n. Each T_n is the light that hits the image given that O_n has reflectance 1 and unit lighting.

The template that represents the expected window when the O_n have reflectances r_n is $\sum_n r_n T_n$. Hence the likelihood of the window when the objects have reflectances r_n is shown in equation 10.

$$P(W|\mathbf{C}\&r(O_1)=r_1 \cdots r(O_n)=r_n\&M) = \kappa \exp(-(W - \sum_n r_n T_n)(W - \sum_n r_n T_n)^T/2V) \tag{10}$$

To derive the likelihood of the window when the objects are of unknown reflectance it is necessary to integrate over all possible sets of r_n. Equation 11 shows the integrated equation.

$$P(W|\mathbf{C}\&M) = \kappa \int \exp(-(W - \sum_n r_n T_n)(W - \sum_n r_n T_n)^T/2V) dD_\epsilon r_n \tag{11}$$

I use the simplifying notation from section 6.1. Also let C_n be WT_n^T. Then equation 11 can be transformed into equation 12.

$$P(W|\mathbf{C}\&M) = \kappa \exp(-W^2 + w\alpha^2/2V) \int \exp(\sum_n r_n C_n/V) \exp(-(\sum_n r_n T_n)(\sum_n r_n T_n)^T/2V) dD_\epsilon r_n \tag{12}$$

As in section 6.1 I can break up equation 12 into a pair of functions F_1 and F_4. F_1 is as before F_4 is described by equation 13.

$$F_4(C_1, C_2, \cdots C_{N_O}) = \int \exp(\sum_n r_n C_n/V) \exp(-(\sum_n r_n T_n)(\sum_n r_n T_n)^T/2V) dD_\epsilon r_n \tag{13}$$

Unlike F_2, F_4 is a function of several variables. Thus using a table for table lookup on F_4 takes a large amount of memory since an entry must be made available for each possible value of the elements. Experiments with constructing F_4 tables for 2 object configurations show that F_4 is smooth and near quadratic. As an example F_4 for standard deviation 12 noise and 5 by 5 templates is plotted in figure 9.

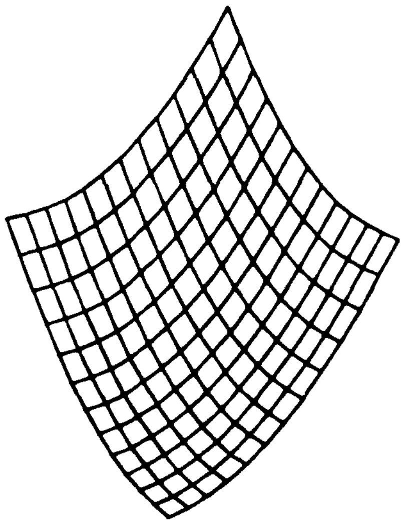


Figure 9: log(F_4) with stdev 12 noise and 5x5 template

Hence the tables can be stored sparsely and splines or even linear interpolation used to get at values that were not given without introducing serious inaccuracy. If the table is close enough to a quadratic function then perhaps only a polynomial need be stored.

If all the T_n are orthogonal (their pairwise vector products are 0) and D_e is a probability distribution where colors of objects are uncorrelated then F_4 can be partitioned into a product of N_O functions. Each of these functions computes the probability that the relevant part of the window observed data is the image of O_n for some n. Here N_O large precise tables can be stored to compute each of the functions and their product used for F_4.

Since F_4 can be calculated there is a feasible algorithm for determining the likelihood of an observed image given a particular pair of objects. A description of that algorithm and the times spent in each step is shown in figure 10.

Figure 10: Algorithm to Calculate Likelihood of Multiple Objects for a Single Window

			Multiplies	Adds
W^2	:=	WW^T	w	$w-1$
N_O times C_i	:=	WT_1^T	$N_O w$	$N_O w - N_O$
Output		$F_1(W^2)^* F_4(C_1,C_2)$	1	0

I did not consider the cost of the interpolation or splining for F_4 in the operation counts in figure 10. The cost of this algorithm is $(N_O+1)w+1$ multiplies and $(N_O+1)(w-1)$ adds plus the cost incurred by the interpolation. This cost occurs for each window.

6.4. Blurred Images

The previous sections assume that the only degradation of the image data is a result of adding a normal random variable to each element of the image (noise axiom). However lenses and many other sensors degrade the image through blur. Motion also causes blur on film and other sensors. Blur is often modeled as a linear transformation of the image data that is applied before the normal variable is added to the pixels [Andrews77]. When blur is shift independent (same blur everywhere in the image) then the linear operator must be a convolution operator.

The algorithms specified in sections 6.1 and 6.3 require changes to work under this new assumption. If the blur is linear the change required to these algorithms is to use blurred templates. The likelihood of the observed data given a template is a function of the vector difference between the expected observation without the normal additive iid and the observed data. If it is expected that a blurring has happened then one need to use a blurred template instead of the unblurred template used in the previous algorithms.

The algorithm of section 6.3 uses two templates (one for each object). If the blur is linear then the linear function aT_1+bT_2 and the blur function commute. Thus the two templates can be blurred and the result of correlating the window with the two blurred templates can be used with this algorithm.

The problem I address in the rest of this section is how to compute a blurred template from an unblurred template. A shift independent blurring operation is applying a convolution operator to the image. A convolution operator has limited extent if the illuminance of at a pixel in the blurred image depends on a window in the unblurred image. Such a convolution operator can be described by a matrix the size of the window that describes the effect each point in the window has on that point of the blurred image. So a blurring function that causes each point of the blurred image to be the result of .5 from the corresponding point of the unblurred image and .25 from the points immediately to the right and the left has a matrix described by figure 11.

.25	.5	.25

Figure 11: Simple Blurring Function Matrix

Given a blurring function matrix of size (M_w, M_l) and an unblurred template of size (T_w, T_l), a blurred template of size $(T_w - M_w + 1, T_l - M_l + 1)$ can be calculated (figure 12) ($\otimes$ means convolution).

T:

100	100	100	200	200	200

$\otimes$

M:

.25	.5	.25

=

$T \otimes M$:

100	125	175	200

Figure 12: Effect of Blur Matrix M on Template T

To develop a larger blurred template than $(T_w - M_w + 1, T_l - M_l + 1)$ requires that the blur function be applied to points outside the unblurred template. If the expected values for such points are derived then a larger unblurred template has been constructed. Hence the derivation of a smaller blurred template from an unblurred template suffices for the construction of blurred templates.

6.5. Correlated Noise

The previous sections assume that an uncorrelated normal variable called noise that is added into the illuminance of each pixel in the ideal image to get the observed image. It is possible to relax the assumption that the noise added to each pixel is uncorrelated with the noise added to the other pixels. Instead a matrix C can be supplied that describes the correlation between the noise variables of the window.

One problem is how to handle correlations between points in the window and points outside the window. Since one can only correlate with expected values of points outside the window (since we chose to ignore the data from such points in our calculations) the effect of such points can only introduce a constant factor into the likelihood calculations. When the likelihoods are converted into probabilities this constant factor is divided out. Hence I can safely ignore such a constant factor. For the purposes of evidence theory I may need to derive the constant factor but it need only be derived once and then all the likelihoods be only multiplied by it.

The algorithm in section 6.3 has the algorithm in sections 6.1 as a special case. Thus if I derive the algorithm corresponding to the one in section 6.3 I can derive the other algorithm. If I have window W and I expect (possibly blurred) templates T_n with unknown reflectances then the equation that describes the likelihood of W is equation 14.

$$P(W|O_1 \cdots O_n \& M) = \kappa \int \exp(-(W - \sum_n r_n T_n) C (W - \sum_n r_n T_n)^T / 2) dD_\epsilon r_n \tag{14}$$

I introduce notation to simplify equation 14 to the point where an algorithm naturally derives from it. Let W_C^2 be WCW^T. C is symmetric so let $c_n = WCT_n^T = T_1 CW^T$. Then equation 14 can be rearranged and simplified to equation 15.

$$P(W|O_1 \cdots O_n \& M) = \kappa \exp(-W_C^2/2) \int \exp(\sum_n r_n c_n) \exp(-(\sum_n r_n T_n) C (\sum_n r_n T_n)^T / 2) dD_\epsilon r_n \tag{15}$$

I can then describe equation 15 as the product of two functions, F_5 that takes W_C^2 as an argument and F_6 that takes the set of c_n as arguments. Equations 16 describe F_5 and F_6.

$$\begin{aligned} F_5(X) &= \kappa \exp(-X/2) \\ F_6(X_1, \cdots X_{N_O}) &= \int \exp(\sum_n r_n X_n) \exp(-(\sum_n r_n T_n) C (\sum_n r_n T_n)^T / 2) dD_\epsilon r_n \\ P(W|O_1, \cdots O_{N_O} \& M) &= F_5(W_C^2) F_6(c_1, \cdots c_{N_O}) \end{aligned} \tag{16}$$

Equation 16 is simple enough to derive an algorithm that calculates the likelihood of a window given a template and correlated noise with standard deviation σ and correlation matrix C. Figure 13 shows this algorithm.

Figure 13: Algorithm to Calculate Likelihood of Multiple Objects with Correlated Noise

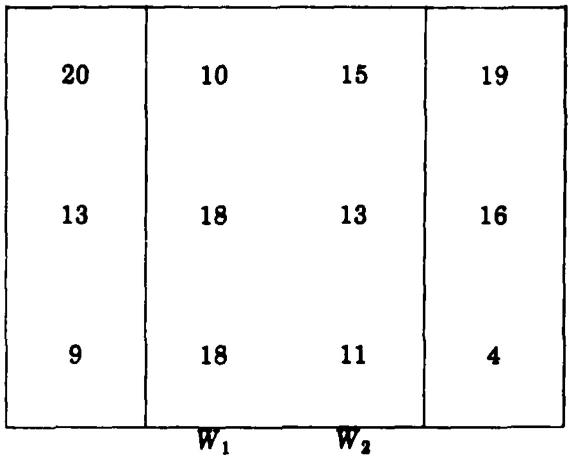
			Multiplies	Adds
W_C^2	:=	WCW^T	$w(w+1)$	$(w+1)(w-1)$
N_O times c_2	:=	WCT_n^T	$N_O w$	$N_O(w-1)$
Output		$F_5(W_C^2)$*$F_6(c_1,c_2)$	1	0

Like F_4, F_6 may require interpolation. The cost of the potential interpolation was not figured into these calculations. The algorithm with correlation between the noise variables requires $w(w+1)+N_O w+1$ multiplies and $(w+1)(w-1)+N_O(w-1)$ adds. Substantial savings may be found when C is sparse. Correlation matrices are typically band matrices. If there are b bands in C then the number of multiplies is less than $(b+1)w+N_O w+1$ and the number of adds is less than $(b+1)(w-1)+N_O(w-1)$ adds.

6.6. Sharing the Work

The algorithms in figures 8, 10 and 13 are algorithms for finding the likelihood of a particular template for a single window. Many of an observed image's windows overlap. If likelihoods are being computed for two overlapping windows much of the work in computing the likelihoods can be shared between the computations on the two windows. If the likelihoods are being computed for every window on the image such savings can be substantial.

When taking the sum of the elements of two overlapping windows, as is one in the algorithm of figures 8 it is necessary to only sum the overlap once. Figure 14 gives an example of this savings.



20	10	15	19
13	18	13	16
9	18	11	4

$$\Sigma(W_1 \cap W_2) = 10+15+18+13+18+11 = 85$$
$$\Sigma(W_1) = 20+13+9+\Sigma(W_1 \cap W_2) = 127$$
$$\Sigma(W_2) = \Sigma(W_1 \cap W_2)+19+16+4 = 124$$

Figure 14: Summing the elements of two overlapping windows

The work in summing the squares of the elements in two windows can be shared this way too. If the likelihood generator is being used on every window on an image then the work needed to calculate sums and sum of squares is a fraction of that needed to calculate the same statistics for the same number of non-overlapping windows.

If every window (or a substantial fraction thereof) of an image has the algorithms in figure 8 or 10 run on them then the work involved in convolving the image with templates can be saved too (at least when the templates grow large). Convolution can be performed with the fast Fourier transform at substantial saving in operations for large templates. For algorithm 13 when the correlation matrix has structure (such as being a band matrix) then the fast Fourier transform can be used with substantial savings too.

Thus much of the work can be shared when likelihoods are being determined for every window of an image. Hence the likelihood generators described in figures 8, 10, and 13 are competitive in speed with most standard edge detections schemes.

Another way the work can be shared is that some of the templates used that describe object configurations in a window is by describing the configuration in another template shifted 1 pixel over (see figure 15).

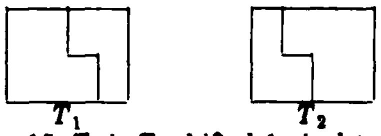


Figure 15: T_1 is T_2 shifted 1 pixel to the right

If every window in the image is being processed then the likelihoods corresponding to the template T_2 are approximated by the likelihoods calculated by the template T_1 for the window 1 pixel to the right. To realize why such an approximation is good, consider that using a window is itself an approximation. The likelihood of the configuration of objects described by T_1 is approximated by running the algorithm over a window. The likelihood of the configuration described by T_2 can be approximated by running the same algorithm over a window shifted 1 to the right.

Thus the likelihoods for the template corresponding to T_2 need only be calculated for the windows on the far right hand side of the image (the other windows have the T_1 template run on them already). Thus instead of having to take into account templates corresponding to the same configuration of objects shifted several pixels in some direction one need only use a single template and use the output from this template on windows shifted in that direction.

6.7. Getting Probabilities from Likelihoods

Given likelihood generators the remaining task is to calculate probabilities from these likelihoods (using priors). The first task that needs to be done is to group the likelihoods generated into sets that support different labelings for the features. Thus if the configurations C_1, C_2, C_3 correspond to the existence of a boundary at a point and C_4, C_5 and C_6 represent situations that aren't boundaries at that point then I must collect the likelihoods based on C_1, C_2 and C_3 into a single likelihood and similar with the likelihoods collected from C_4, C_5 and C_6. Then I would have a likelihood corresponding to each possible labelings of my feature (for boundary point detection I need to determine the likelihoods corresponding to the existence of a boundary and those that correspond to the nonexistence). Given these likelihoods I can use Bayes' rule to derive probabilities (see equation 3).

The likelihood of a boundary is the probability of the observed scene being generated when an object configuration corresponding to the existence of a boundary exists. I can derive an equation for calculating the probability of a boundary from the outputs of my likelihood generators if I have the prior probability that the configuration is the position of the objects in the scene for each configuration that corresponds to a boundary. Let the set of configurations that correspond to a boundary be represented by $\mathbf{C}_b$. Equation 17 is the first step in the derivation expanding out the likelihood into conditional probabilities.

$$P(W_O|\mathbf{C}_b) = \frac{P(W_O\mathbf{C}_b)}{P(\mathbf{C}_b)} \tag{17}$$

Since the real scene can not correspond to two different configurations I can expand equation 17 into equation 18.

$$P(W_O|\mathbf{C}_b) = \frac{\sum_{c\in\mathbf{C}_b} P(W_O \,\&\, c\in\mathbf{C}_b)}{\sum_{c\in\mathbf{C}_b} P(c\in\mathbf{C}_b)} \tag{18}$$

A slight change to equation 18 introduces the likelihoods generated by the algorithms in figures 8 through 13 and the prior probabilities that the scene is in a configuration tested by the algorithms. Equation 19 shows this change.

$$P(W_O|\mathbf{C}_b) = \frac{\sum_{c\in\mathbf{C}_b} P(W_O|c\in\mathbf{C}_b)P(c\in\mathbf{C}_b)}{\sum_{c\in\mathbf{C}_b} P(c\in\mathbf{C}_b)} \tag{19}$$

Equation 19 allows me (given priors on the templates or template sets) to gather many likelihoods into a single one. If I have likelihoods for every feature label and prior probabilities that the feature takes on that label I can use Bayes' law as in equation 3 (reprised here) to derive probabilities for feature labels given the data in the window.

$$P(l_f|O\&M) = \frac{P(O|l_f\&M)P(l_f|M)}{\sum_{l'_f \in L_f} P(O|l'_f\&M)P(l'_f|M)} \qquad (3)$$

6.8. Estimating Boundaries

In section 6.7 I show how to derive probabilities given a likelihood generator. Often one must use programs (e. g. programs supplied as part of a package) that take as input estimates of the positions of the boundaries. Such programs can not use probabilities, they just want a boundary map. Here I show how to generate such an input.

To estimate where the boundaries are in an image it is necessary first to develop a cost function that describes what costs errors in estimation have. To use the probabilities of boundaries at points to estimate the configuration of boundaries in an image optimally it is necessary to use a cost function that sums the effects of pointwise errors. Such cost functions are simple to understand and require few parameters to describe (namely only the costs of different mislabelings at a point). I only use this type of cost function in this part of the paper. I also assume that making a correct decision has 0 cost.

For boundary point detection the costs that need to be calculated are:

(1) the cost of labeling a point as a boundary when there is no boundary there.

(2) the cost of labeling a point as not being a boundary when it is.

Call the cost of labeling a point (x,y) as a boundary point when it isn't $c_1(x,y)$ and the cost of labeling a point as not being a boundary when it is $c_2(x,y)$. Let $p_B(x,y)$ be the probability of a boundary at (x,y). Let $e_B(x,y)$ be 1 when the estimation procedure indicates there is a boundary at (x,y) and 0 otherwise. We want a detector that minimizes the expected cost for the estimation. Thus we want to minimize the summation in equation 20.

$$\sum_{(x,y)} c_1(x,y)(1-p_B(x,y))e_B(x,y) + c_2(x,y)p_B(x,y)(1-e_B(x,y)) \qquad (20)$$

Let us assume that $c_1(x,y)$ is the same for all (x,y) and the same for c_2. Equation 20 is clearly minimized by minimizing equation 21 for each (x,y) ((x,y) is deleted for clarity).

$$c_1(1-p_B)e_B + c_2p_B(1-e_B) \qquad (21)$$

Equation 21 can be rearranged into equation 22.

$$(c_1(1-p_B) - c_2p_B)e_B(x,y) + c_2p_B \qquad (22)$$

Clearly you want e_B to be 1 when equation 23 is positive and e_B to be 0 when equation 23 is negative.

$$c_1(1-p_B) - c_2p_B \qquad (23)$$

This statement can be algebraically transformed into the statement that e_B should be 1 when the inequality in equation 24 is satisfied and 0 otherwise.

$$p_B < \frac{c_1 - c_2}{c_1} \tag{24}$$

Thus one only needs to threshold the probabilities of boundaries with $\frac{c_1 - c_2}{c_1}$ to estimate the positions of the boundary for the additive cost function with costs c_1 and c_2. This argument is standard in Bayesian decision theory with simple loss functions [Berger80].

7. Implementation Details

Here, I describe my implementation of the algorithms described in section 6. I have code for the algorithms in figures 8, and 10. I also have constructed the code that is implied by equations 19 and 3 of section 6.7.

7.1. Likelihood Generators

These algorithms are based on the assumption that the scene can be modeled by a set of templates. The templates are objects of unit reflectance under unit lighting. I have one template that represents the window being in the interior of the object shown in figure 16.

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

Figure 16: Template for the Interior of an Object

For each of 4 directions, 0 degrees, 45 degrees, 90 degrees, and 135 degrees, I have 3 pairs of templates that describe three possible boundaries. For example figure 17 shows the 0 degree templates.

1^{st} pair

1	1	0.5	0	0
1	1	0.5	0	0
1	1	0.5	0	0
1	1	0.5	0	0
1	1	0.5	0	0

0	0	0.5	1	1
0	0	0.5	1	1
0	0	0.5	1	1
0	0	0.5	1	1
0	0	0.5	1	1

2^{nd} pair

1	1	0	0	0
1	1	0	0	0
1	1	0	0	0
1	1	0	0	0
1	1	0	0	0

0	0	1	1	1
0	0	1	1	1
0	0	1	1	1
0	0	1	1	1
0	0	1	1	1

3^{rd} pair

1	1	1	0	0
1	1	1	0	0
1	1	1	0	0
1	1	1	0	0
1	1	1	0	0

0	0	0	1	1
0	0	0	1	1
0	0	0	1	1
0	0	0	1	1
0	0	0	1	1

Figure 17: Template for the 0 degree Boundary

Each pair of templates represent two objects, one occluding the other. I have not generated any templates for windows with 3 objects.

The algorithms in figures 8, and 10 consist of a part that is dependent on a template used and a part that is a function of the observed window. It is the part that depends on the template that presents implementation difficulties. Function F_2 is a function of the sum of squared elements in the template. Function F_4 is a function of the pairwise products of the templates representing the objects in the configuration. Both of these functions were implemented by table lookup with linear interpolation in my implementation. For every direction the sum of squares of the templates and the pairwise products are described by the same 5 numbers. Only two F_4 tables need to be generated.

F_2 and F_4 also depend on the standard deviation of the noise in the image. I call a likelihood generator that assumes a specified standard deviation of noise a likelihood generator *tuned* to that standard deviation of noise. I have likelihood generators tuned to noise with σ equal to 4, 8, 12, and 16.

7.2. Probabilities from Likelihoods

I use equation 19 to gather together the three likelihoods generated from the 3 pairs of templates in each direction into a single likelihood. Thus I have the likelihoods for the four directions that a boundary passes through or next to the center pixel.

I also need the likelihood that there is no boundary near or through the center pixel. To get this likelihood I take the likelihood that the window is in the interior of the object and combine it with likelihoods for central boundaries calculated for the neighboring windows. Thus from the 0 degree boundary likelihoods I use the likelihood from the window one pixel left and right and combine it with the likelihood of a noncentral edge.

Thus I have 4 likelihoods for 4 directed boundary points and one likelihood that represents the likelihood that there is no central edge in the image. I then use Bayes' law from equation 3 to compute the probabilities of these 4 states. I then threshold the probabilities at 0.5 to present the results shown in section 8. Throughout I assumed that the prior probability of a central edge is 0.1 and that this probability is equally distributed in all 4 directions. This prior information is sufficient to apply Bayes' law.

8. Results from Implemented Boundary Detectors

I have implemented the algorithms in sections 6.1 and 6.3 figures 8 and 10. Here I describe the results from testing this detector.

The software I have written is flexible. However I have only constructed templates for a restricted set of configurations. I have templates for a step edge model with the low curvature assumption, 4 possible orientations for boundaries, and 255 gray levels in the image. My templates handle boundaries that occur in the center of pixels or between pixels. I have built the templates for a 5x5, 7x7 and 9x9 windows. I also have constructed tables to compute F_1 and F_4 for each of these windows that assume noise of standard deviations 4,8,12, and 16.

8.1. Results with Artificial Images

I have applied these operators to test images constructed by a package of graphics routines. This package was written by Myra Van Inwegen and is described in an upcoming technical report.

I describe my operators applied to two test images generated by this package. One is an image of two circles shown on the left in figure 18a.

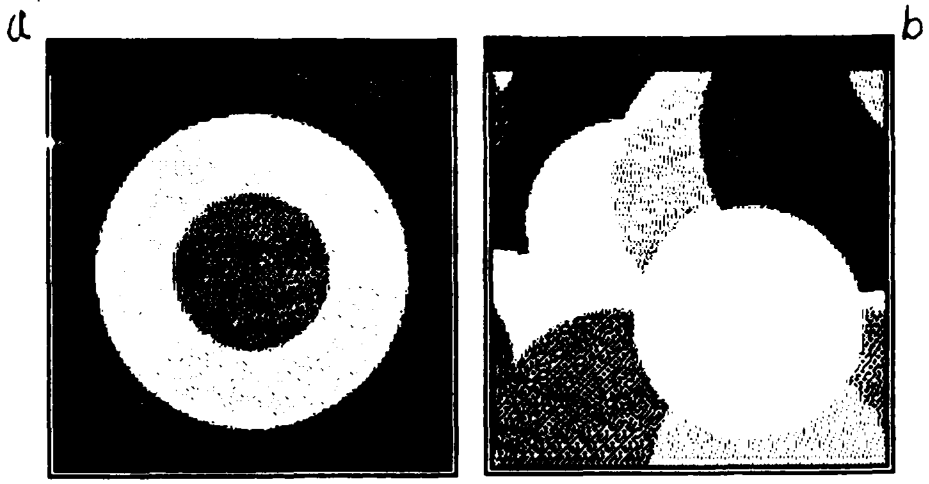
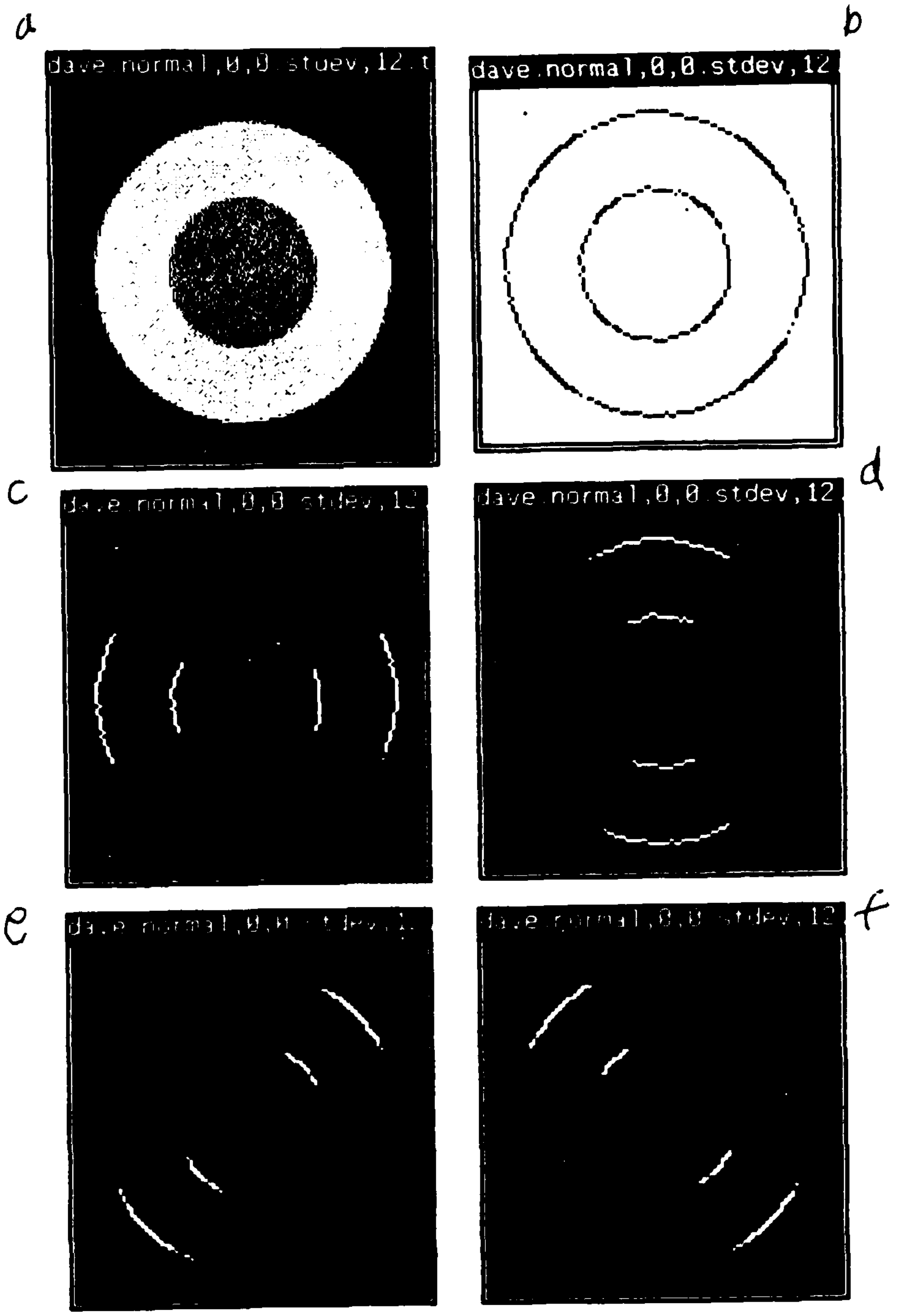


Figure 18: Artificial Images

A more challenging and complex image has also been tested shown in figure 18b.

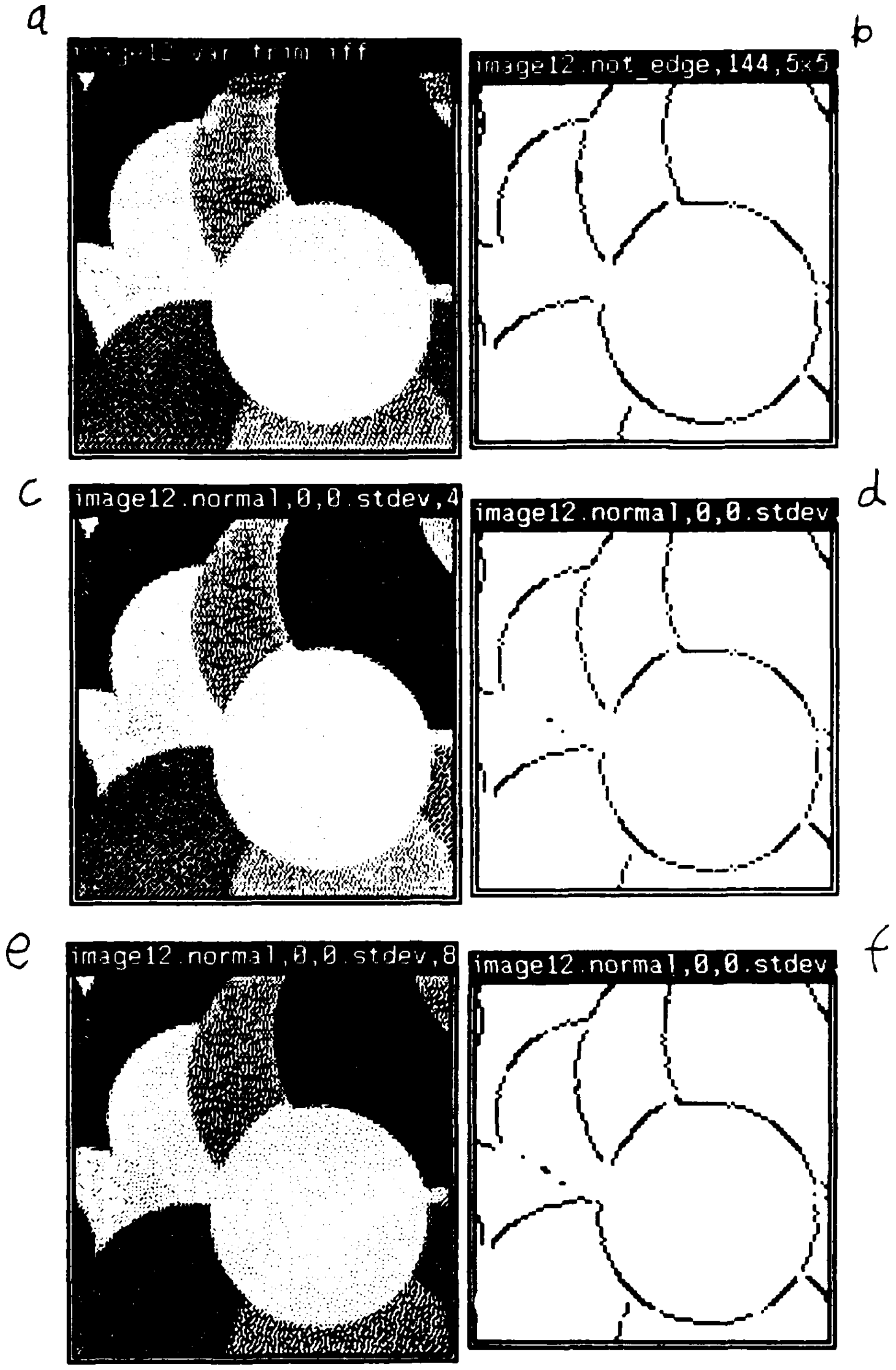
The two circle image (figure 18a) is a particularly good image to test the effect of boundary orientation, curvature and contrast on boundary detection. Figure 19 shows the result of using a 5x5 operator tuned to standard deviation 12 noise on image 18a with standard deviation 12 noise added to it. The images are white at points of greater than 50% probability and black at points less than 50% probability.



a: image with stdev 12 noise
b: white if there is no edge
c: white if 0 degree edge
d: white if 90 degree edge
e: white if 45 degree edge
f: white if 135 degree edge

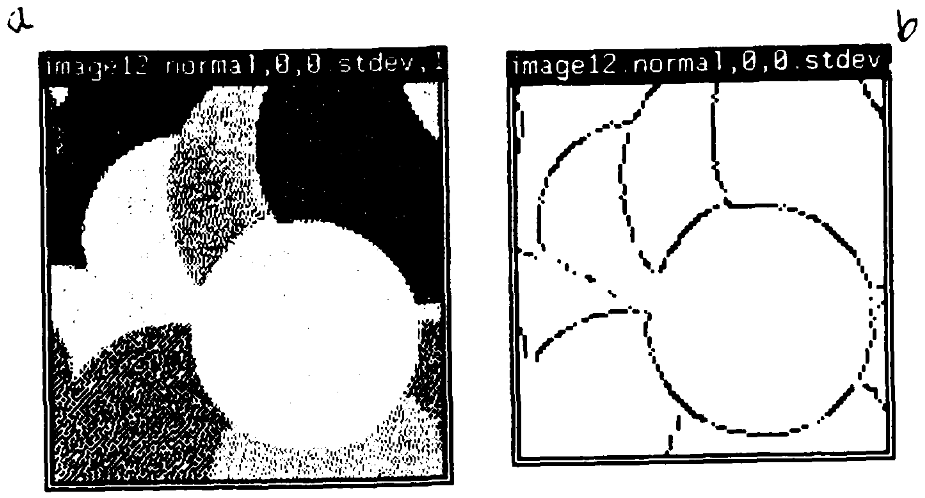
Figure 19: Oriented response for 5x5 operator

In figures 20, 21, and 22 I apply the standard deviation 12 operator to the image 18b with too little, just right and too much noise respectively. The operator output is black when there is greater than 50% probability of a boundary. Note that with too little noise the detector misses boundaries that are there. With too much noise the detector detects boundaries that are not there.



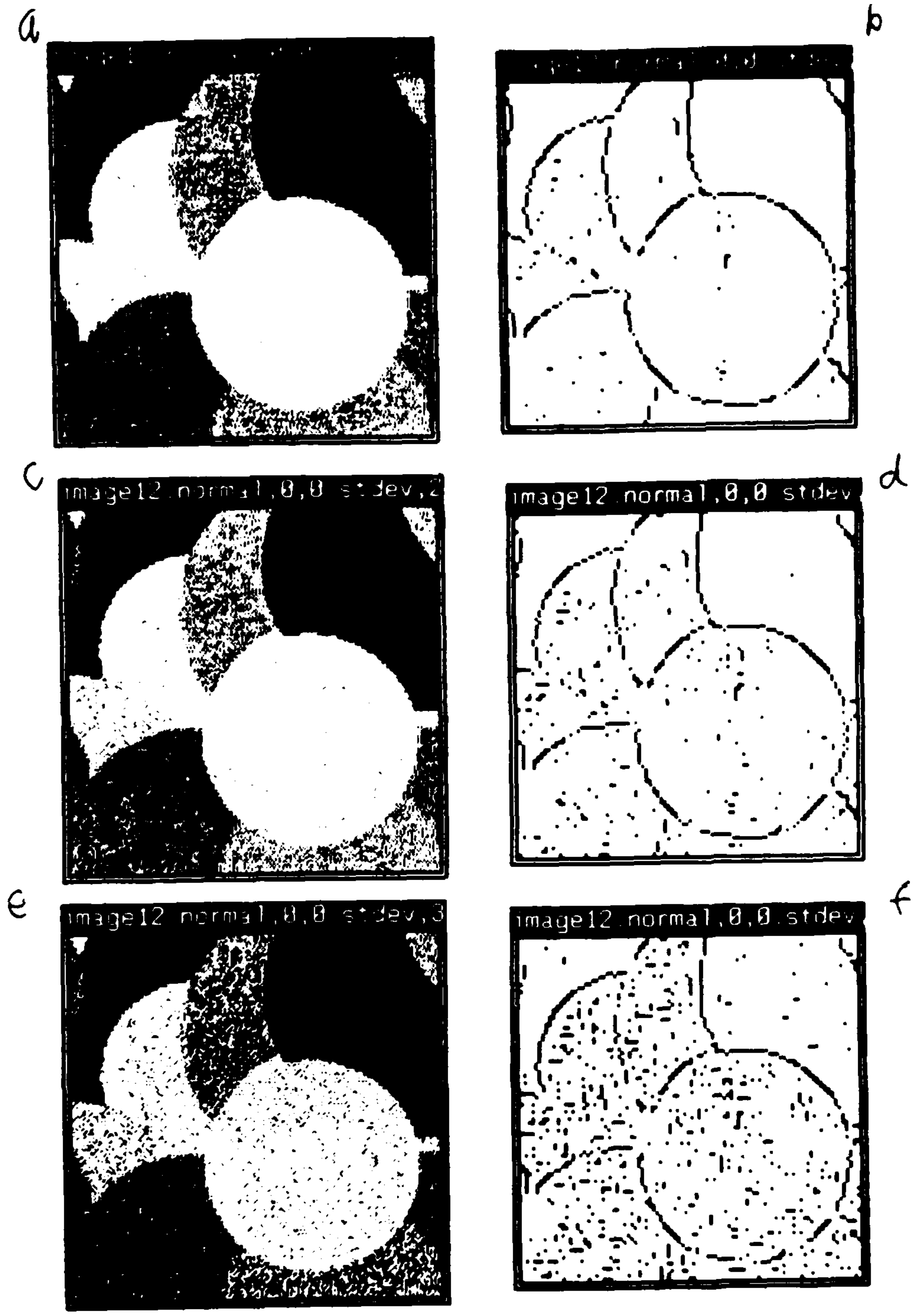
a: image with $\sigma=0$ noise
b: $\sigma=12$ operator on image with $\sigma=0$ noise
c: image with $\sigma=4$ noise
d: $\sigma=12$ operator on image with $\sigma=4$ noise
e: image with $\sigma=8$ noise
f: $\sigma=12$ operator on image with $\sigma=8$ noise

Figure 20: $\sigma=12$ 5x5 operator applied to images with too little ($\sigma<12$) noise



a: image with $\sigma=12$ noise b: $\sigma=12$ operator on image with $\sigma=12$ noise

Figure 21: $\sigma=12$ 5x5 operator applied to image with correct ($\sigma=12$) amount of noise



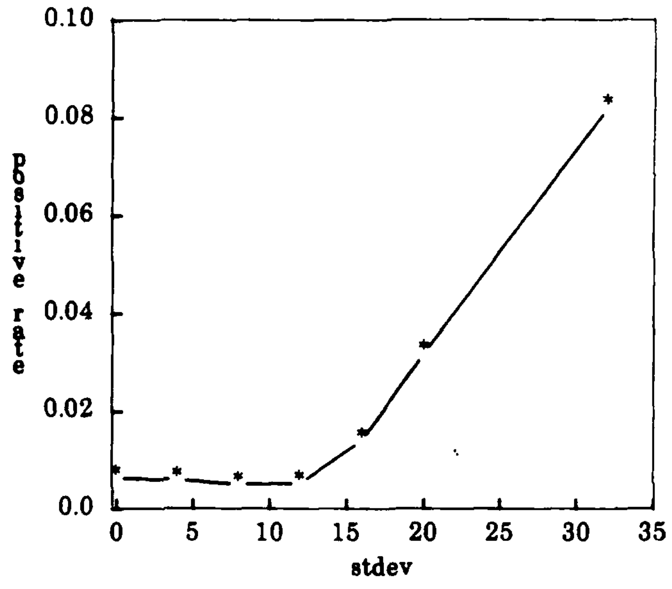
a: image with $\sigma=16$ noise b: $\sigma=12$ operator on image with $\sigma=16$ noise
c: image with $\sigma=20$ noise d: $\sigma=12$ operator on image with $\sigma=20$ noise
e: image with $\sigma=32$ noise f: $\sigma=12$ operator on image with $\sigma=32$ noise
Figure 22: $\sigma=12$ 5x5 operator applied to images with too much ($\sigma>12$) noise

Because these are artificial images I know exactly where the boundaries are. Thus I have written software that counts how many mistakes (false positives and negatives) are made in boundary determination. False negatives are when a boundary that is in the image is missed. False positives are boundaries that are reported where there is no boundary there.

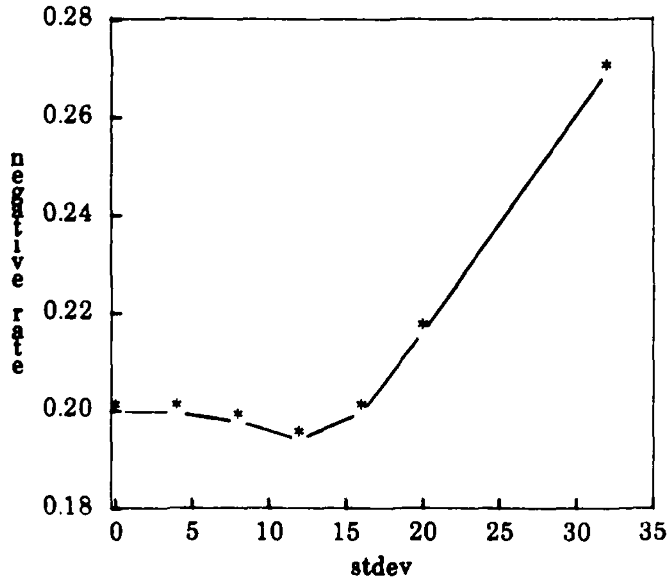
One tricky point is that multiple reports of boundaries are usually considered a bad result [Canny83]. However systems that report a boundary only once will usually have a high false negative rate because they report an edge one pixel off from where it really is. I consider this error to have low enough cost to be ignored. So my software ignores false negatives that are next to reported boundaries in a direction normal to the boundary.

Figure 23 charts the performance of my operator tuned to $\sigma=12$ on the images shown previously. Figure 24 charts the performance of my operator tuned to $\sigma=4$ noise. Figure 25 charts the performance of an operator that is tuned to the same noise level as is contained in the image. Figure 26 superimposes the three graphs of total error rates to show the relationship.

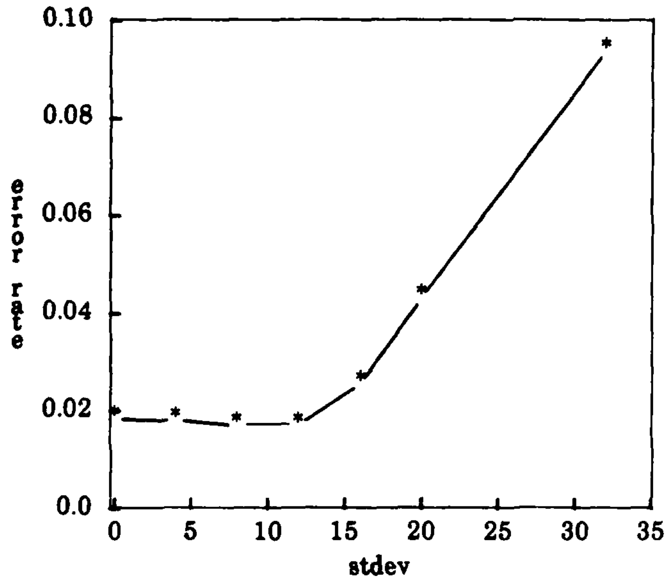
(a)



(b)

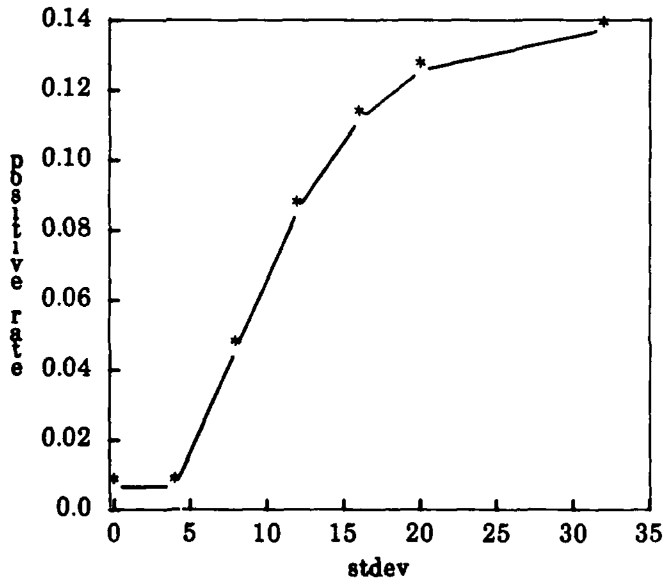


(c)

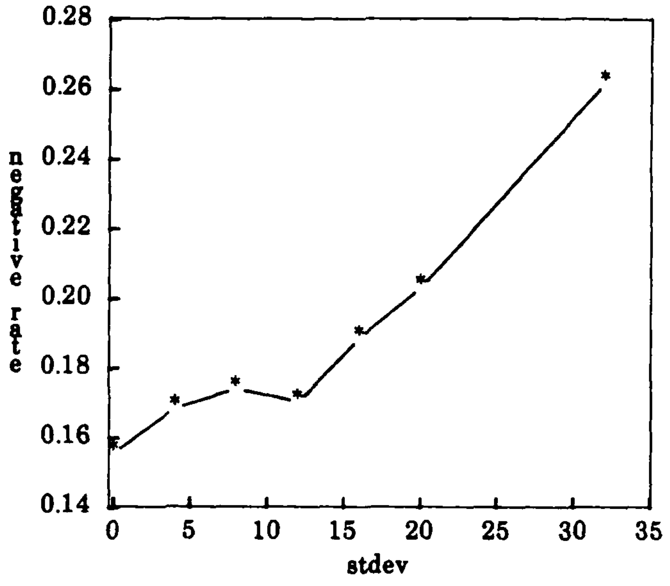


(a) false positive rate vs increasing σ of noise in image
(b) false negative rate vs increasing σ of noise in image
(c) total error rate vs increasing σ of noise in image
Figure 23: Error Rates for the Operator Tuned to $\sigma=12$ noise

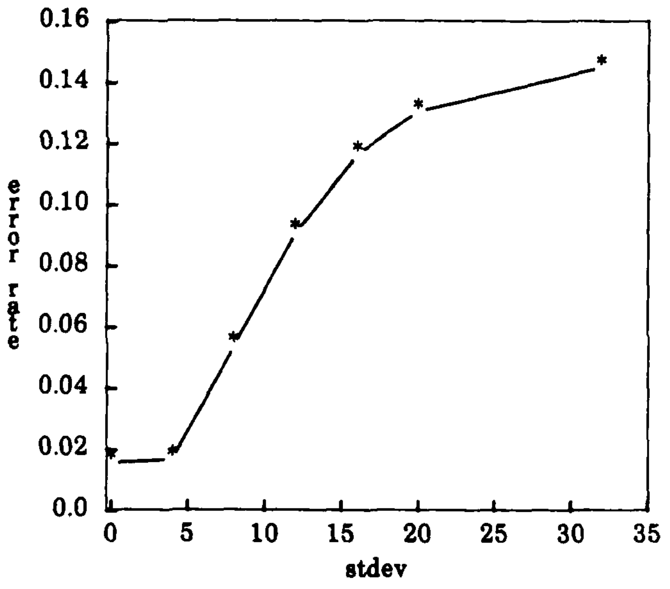
(a)



(b)

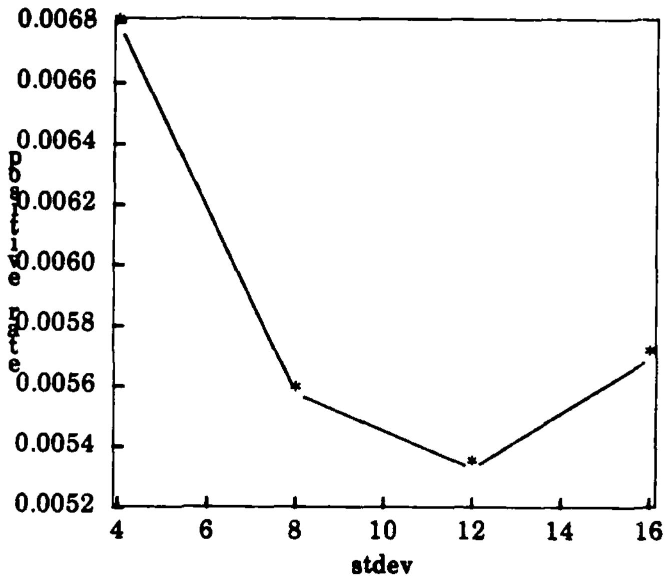


(c)

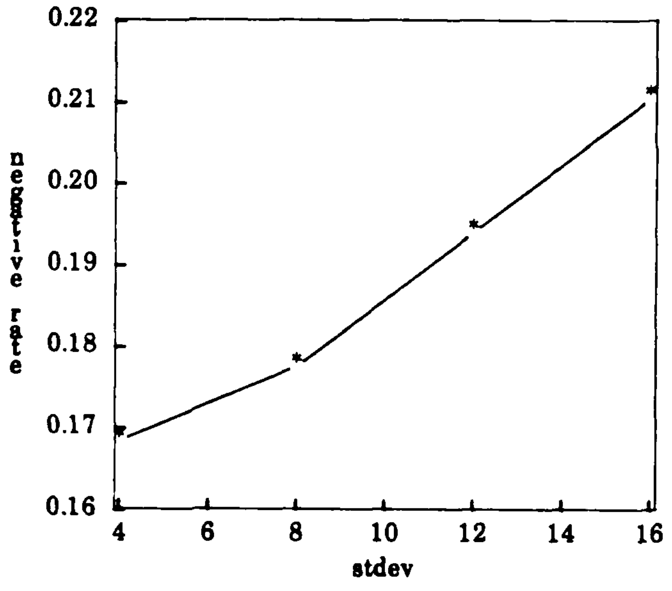


(a) false positive rate vs increasing σ of noise in image
(b) false negative rate vs increasing σ of noise in image
(c) total error rate vs increasing σ of noise in image
Figure 24: Error Rates for the Operator Tuned to $\sigma = 4$ noise

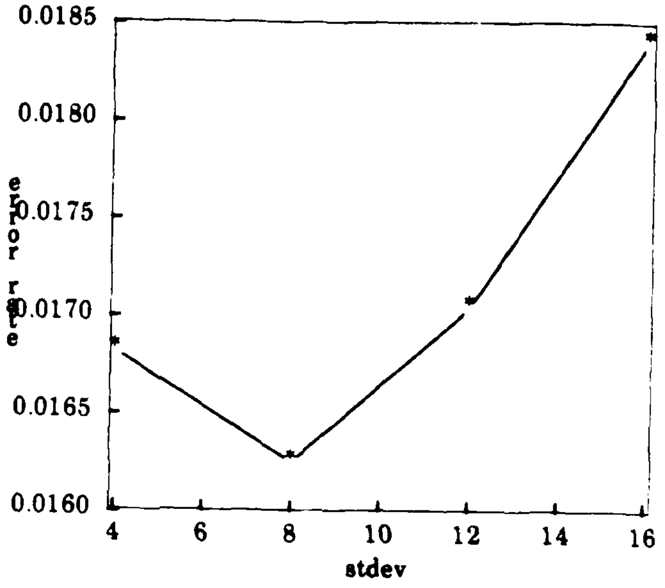
(a)


0.0068
0.0066
0.0064
0.0062
0.0060
0.0058
0.0056
0.0054
0.0052
positive rate
4
6
8
10
12
14
16
stdev

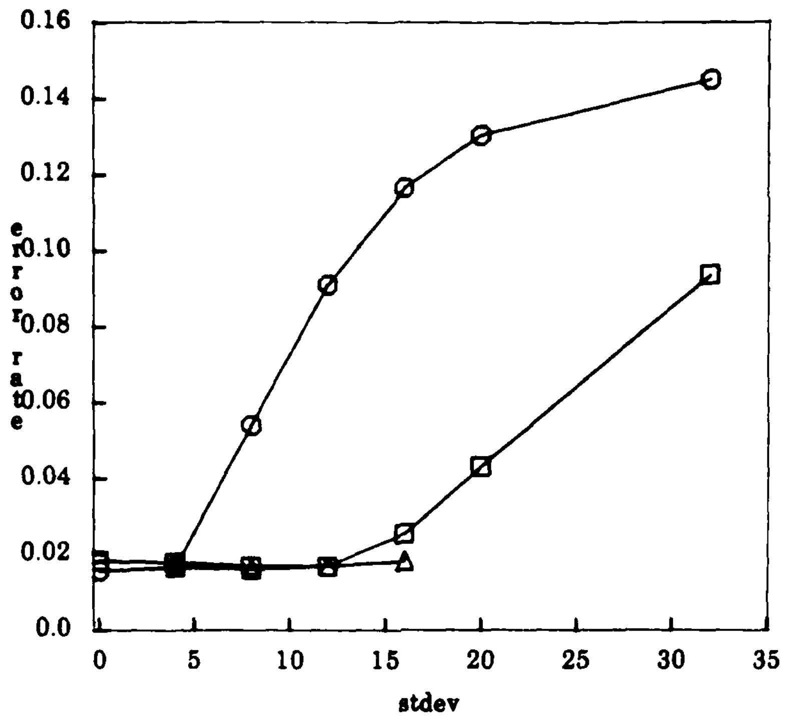
(b)


0.22
0.21
0.20
0.19
0.18
0.17
0.16
negative rate
4
6
8
10
12
14
16
stdev

(c)



(a) false positive rate vs increasing σ of noise in image
(b) false negative rate vs increasing σ of noise in image
(c) total error rate vs increasing σ of noise in image
Figure 25: Error Rates for the Operator Tuned to σ of the noise in the Image



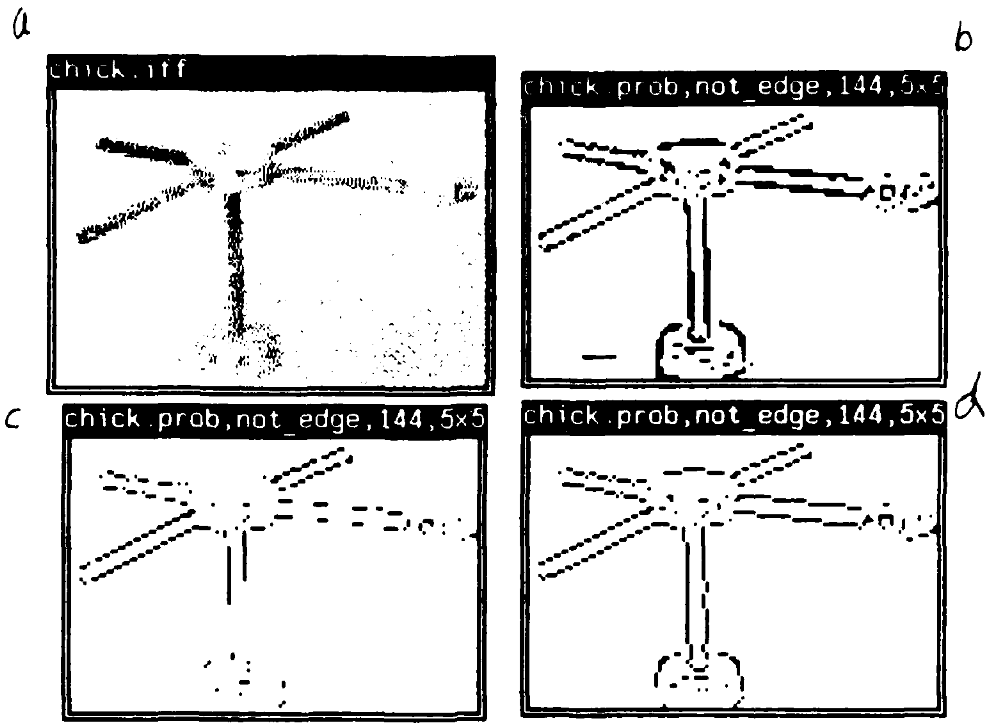
circles: Operator Tuned to $\sigma=12$
squares: Operator Tuned to $\sigma=4$
triangles: Operator Tuned to noise in image

Figure 26: Total error rate for my operators

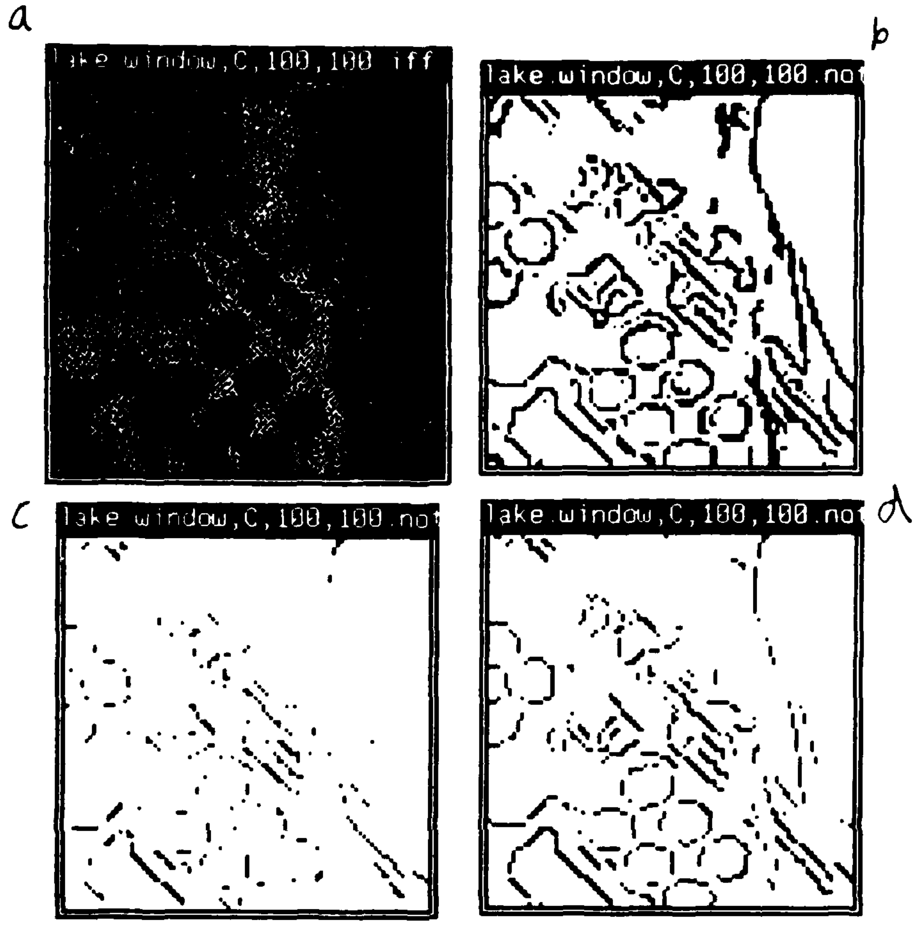
As you can see the tuned operator is always at least as good as the operator that has been developed for stdev 12 or 4 noise.

8.2. Results with Real Images

I have also applied my operator to real images. Here, I use two images, a laboratory image of a Tinkertoy model (figure 27) and an aerial picture (figure 28). I demonstrate the utility of having operators that return probabilities with these results. In both these figures (a) is the image, (b) is the output of my 5x5 stdev 12 operator thresholded at 10% probability, (c) is thresholded at 90% probability and (d) is thresholded at 50% probability. (b) would be used when the cost of missing an edge is high as when the output would be fed to a regularization technique. Note that for case (b) the operator sometimes returns thickened edges. (c) would be used when the cost of missing an edge is low. Hough transform techniques are often developed with that assumption in mind. There is enough information in figure 27c to find the rods of the tinker toy even though all the boundaries are not extant. (d) is what you use when an operator is equally troubled by all errors.


a
chick.tif
b
chick.prob,not_edge,144,5x5
c
chick.prob,not_edge,144,5x5
chick.prob,not_edge,144,5x5
d

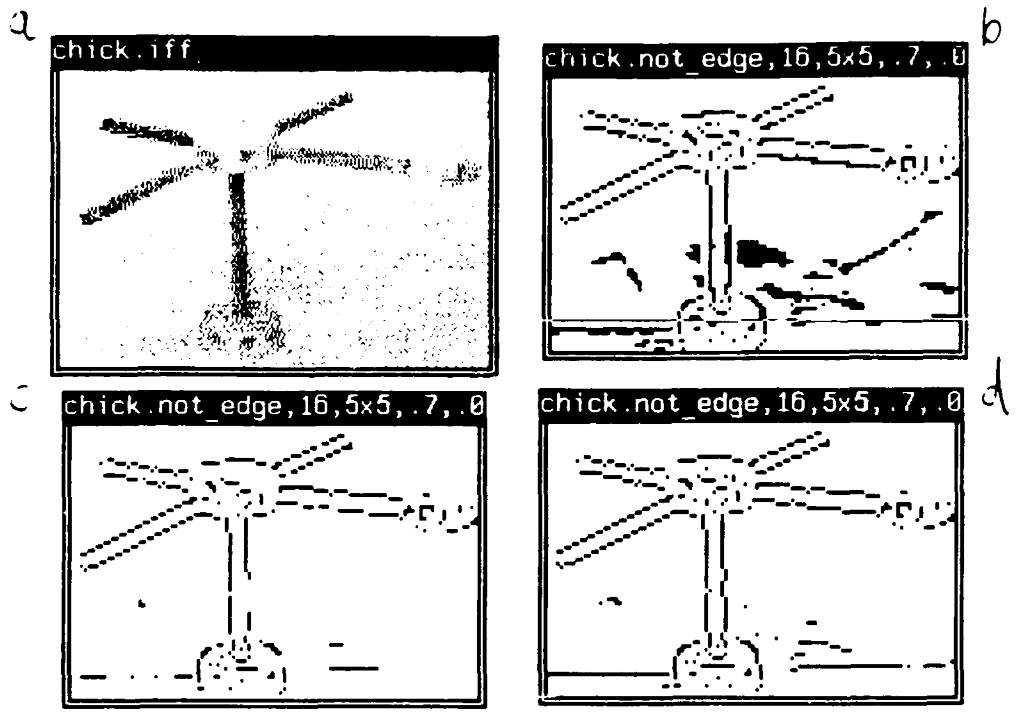
(a) Tinkertoy Image
(b) Output of $\sigma=12$ Operator with threshold at .1
(c) Output of $\sigma=12$ Operator with threshold at .9
(d) Output of $\sigma=12$ Operator with threshold at .5
Figure 27: $\sigma=12$ 5x5 operator applied to tinker toy image



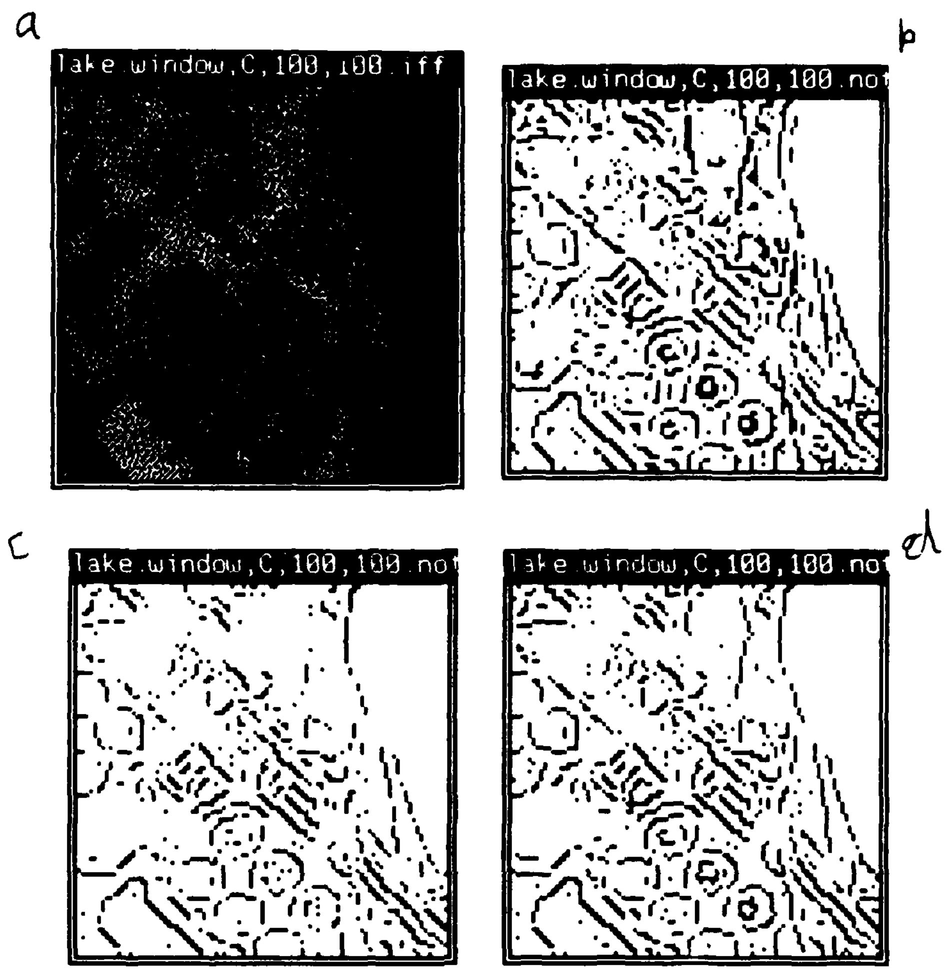
(a) Aerial Image
(b) Output of $\sigma = 12$ Operator with threshold at .1
(c) Output of $\sigma = 12$ Operator with threshold at .9
(d) Output of $\sigma = 12$ Operator with threshold at .5
Figure 28: $\sigma = 12$ 5x5 operator applied to aerial image

A particular threshold may result in a most pleasing (to a human observer) ensemble of results (perhaps .5). But this threshold may not be the best threshold for the succeeding application.

One may notice that lowering the threshold seems to increase the number of boundary points in the regions of real boundaries. It is not surprising that the regions that look most like boundaries should be near boundaries. Also the probabilities of boundaries are being reported as lower than they should be in this image. A good reason for this is that the model used to construct the operator shown is not a good model for this image. In particular there is some evidence [Sher87] that the standard deviation of the noise is closer to 4 than 12 in these images. Thus look at the same results for the $\sigma = 4$ operator in figures 29 and 30.


a
chick.iff
b
chick.not_edge,16,5x5,.7,.0
c
chick.not_edge,16,5x5,.7,.0
chick.not_edge,16,5x5,.7,.0
d

(a) Tinkertoy Image
(b) Output of $\sigma=4$ Operator with threshold at .1
(c) Output of $\sigma=4$ Operator with threshold at .9
(d) Output of $\sigma=4$ Operator with threshold at .5
Figure 27: $\sigma=4$ 5x5 operator applied to tinker toy image



(a) Aerial Image
(b) Output of $\sigma=4$ Operator with threshold at .1
(c) Output of $\sigma=4$ Operator with threshold at .9
(d) Output of $\sigma=4$ Operator with threshold at .5
Figure 28: $\sigma=4$ 5x5 operator applied to aerial image

8.3. Comparisons with Established Techniques

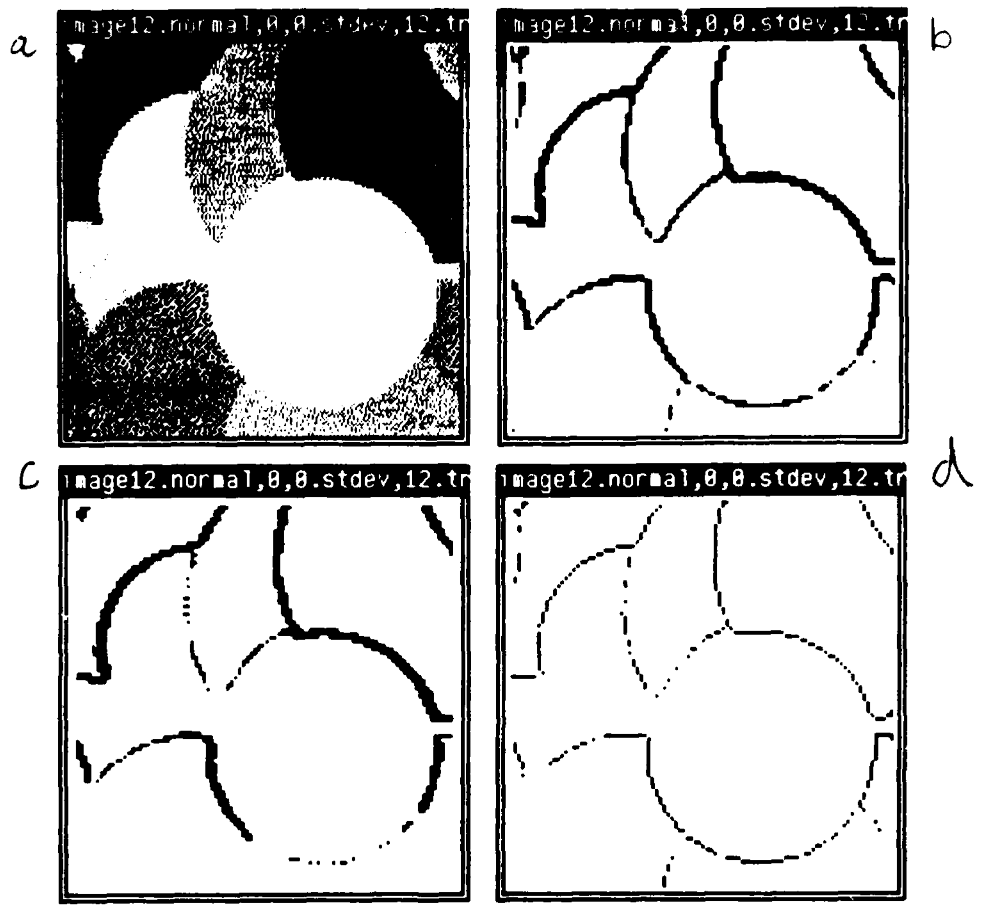
Here, I compare the results I have just presented with established edge detectors. The edge detectors I currently have results for are the Sobel thresholded at 200, the 5 by 5 Kirsch thresholded at 750, and a thinned 3 by 3 Kirsch thresholded at 100. The thresholds for the Sobel and the thinned Kirsch were found to be the ones that minimized the number of errors when applied to the artificial image in figure 18b with standard deviation 12 noise added. The threshold that minimized the errors with the Kirsch was 1175. However at this threshold more than 50% of the boundaries were missed. It was difficult to chart the result of using this operator along with the rest[2] so I used a somewhat lower threshold.

I know these operators are not state of the art. However these results show that tools are available to test this theory against more sophisticated operators. More sophisticated operators

[2]Visual inspection indicated that lowering the threshold would make the result of the operator more appealing. Not that it should matter but ...

such as Canny's and Haralick's will be tested against my operators in the next month or two.

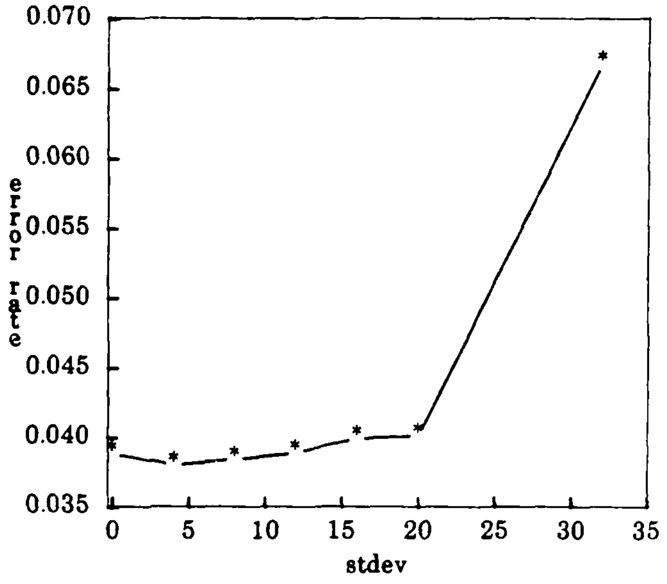
In figure 31 the results of applying the Sobel, Kirsch and thinned Kirsch to my artificial image with stdev 12 noise.


a
b
c
d
image12.normal,0,0.stdev,12.tr
image12.normal,0,0.stdev,12.tr
image12.normal,0,0.stdev,12.tr
image12.normal,0,0.stdev,12.tr

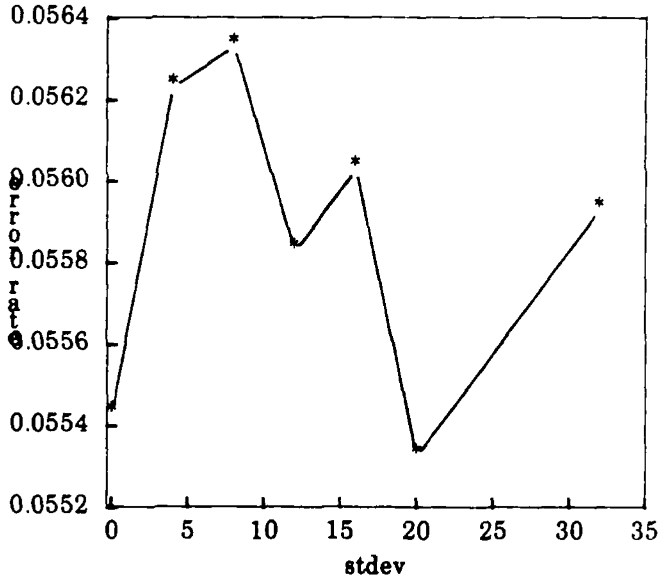
(a) Image with $\sigma = 12$ noise.
(b) Output of the Sobel Optimally Thresholded
(c) Output of the 5x5 Kirsch Optimally Thresholded
(d) Output of the Thinned Kirsch Optimally Thresholded
Figure 31: Application of Established Operators to an Artificial Image

I have measured the error rates for these operators and have charted the total error rates in figure 32.

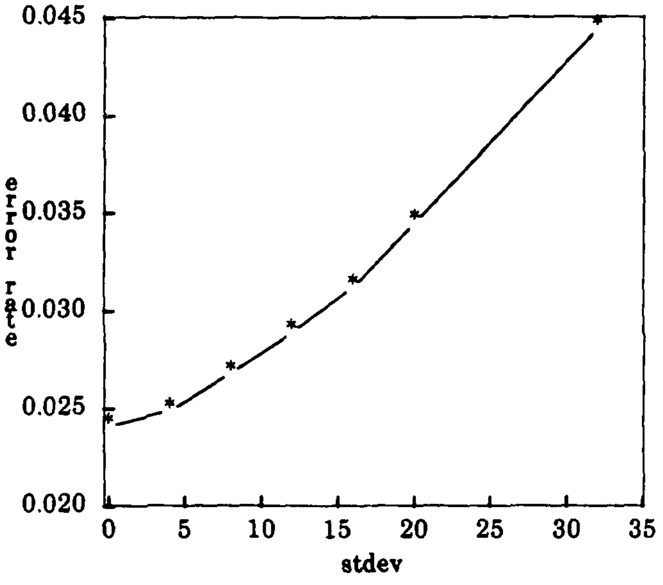
(a)


0.070
0.065
0.060
0.055
0.050
0.045
0.040
0.035
error rate
0
5
10
15
20
25
30
35
stdev

(b)


0.0564
0.0562
0.0560
0.0558
0.0556
0.0554
0.0552
error rate
0
5
10
15
20
25
30
35
stdev

(c)



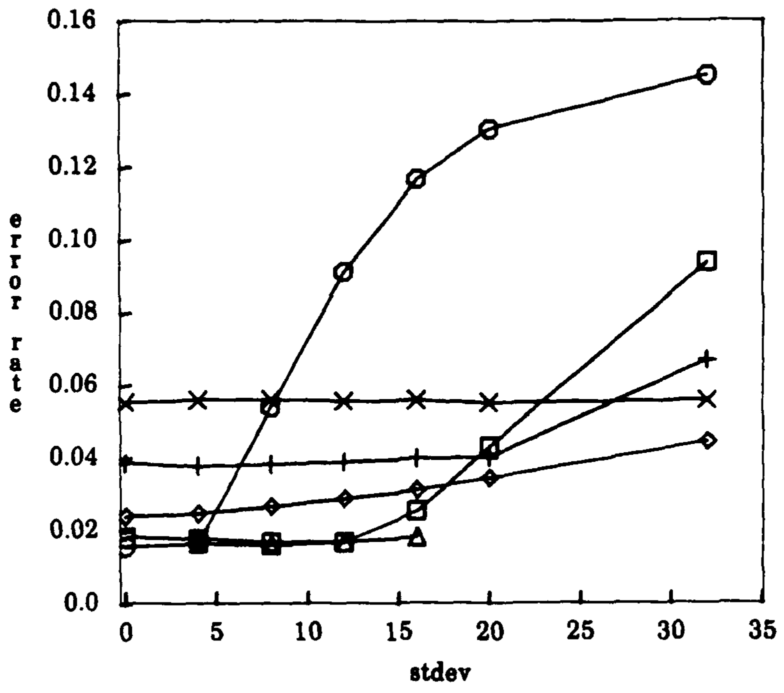
(a) Error rates for the thresholded Sobel

(b) Error rates for the thresholded Kirsch

(c) Error rates for the thinned Kirsch

Figure 32: Charts of Total Error Rates for Established Operators

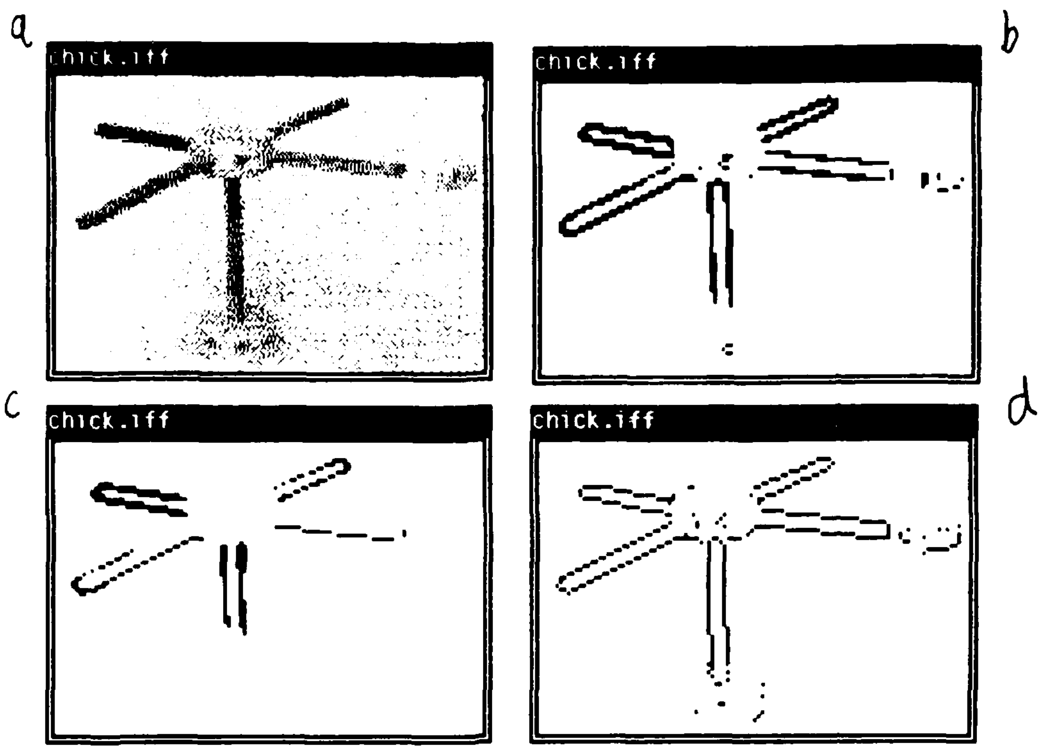
To summarize the results I have a plot that shows the error rates for all the operators I have tested so far (figure 33). In this chart my stdev 12 operator is shown as squares, my tuned operator is shown as circles, the Sobel is shown as triangles, the thresholded Kirsch is shown as crosses and the thinned Kirsch is shown as X's.



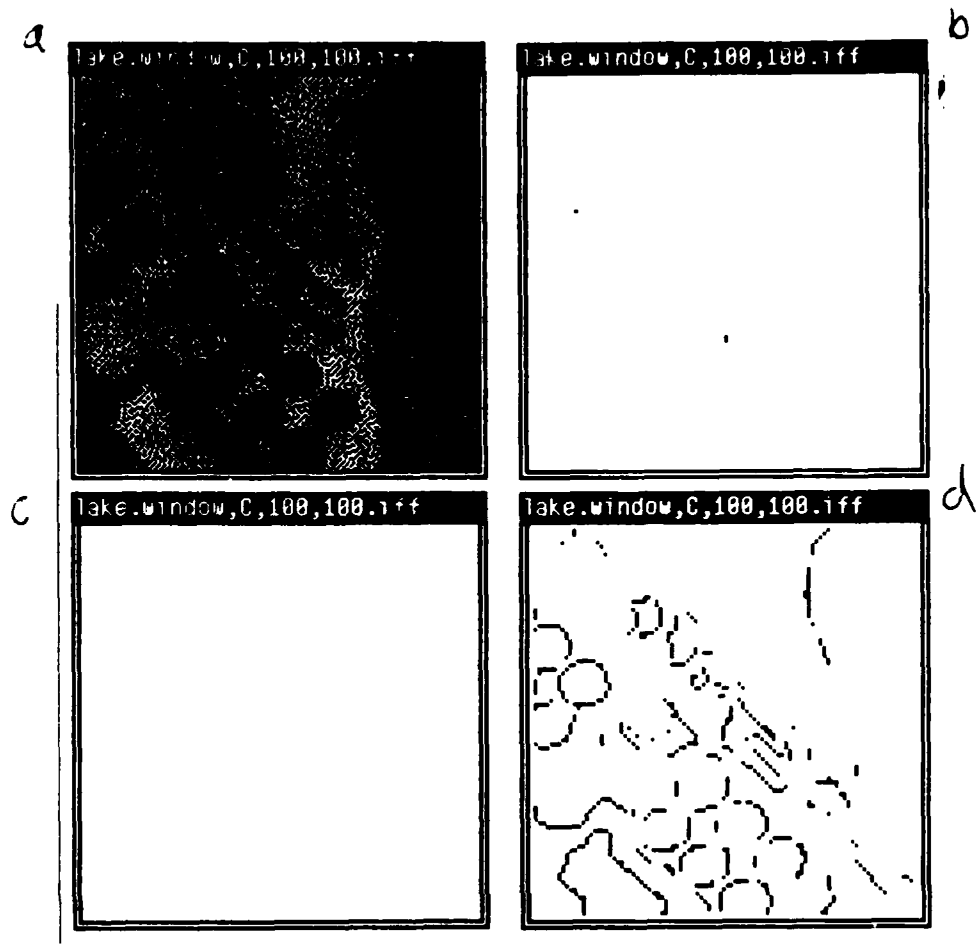
circles:	Tuned $\sigma=12$ operator
squares:	Tuned $\sigma=4$ operator
triangles:	Tuned to noise σ operator
crosses:	Sobel's error rate
X's:	5x5 Kirsch's error rate
diamonds:	Thinned Kirsch's error rate

Figure 33: Error Rates for all Operators

For comparison, I have run the 3 established edge detector on the two real images shown in section 8.2. Figures 34 and 35 show the result of runing these operators with my established operators. Clearly, in these circumstances the most effective established operator for these images is the thinned Kirsch.



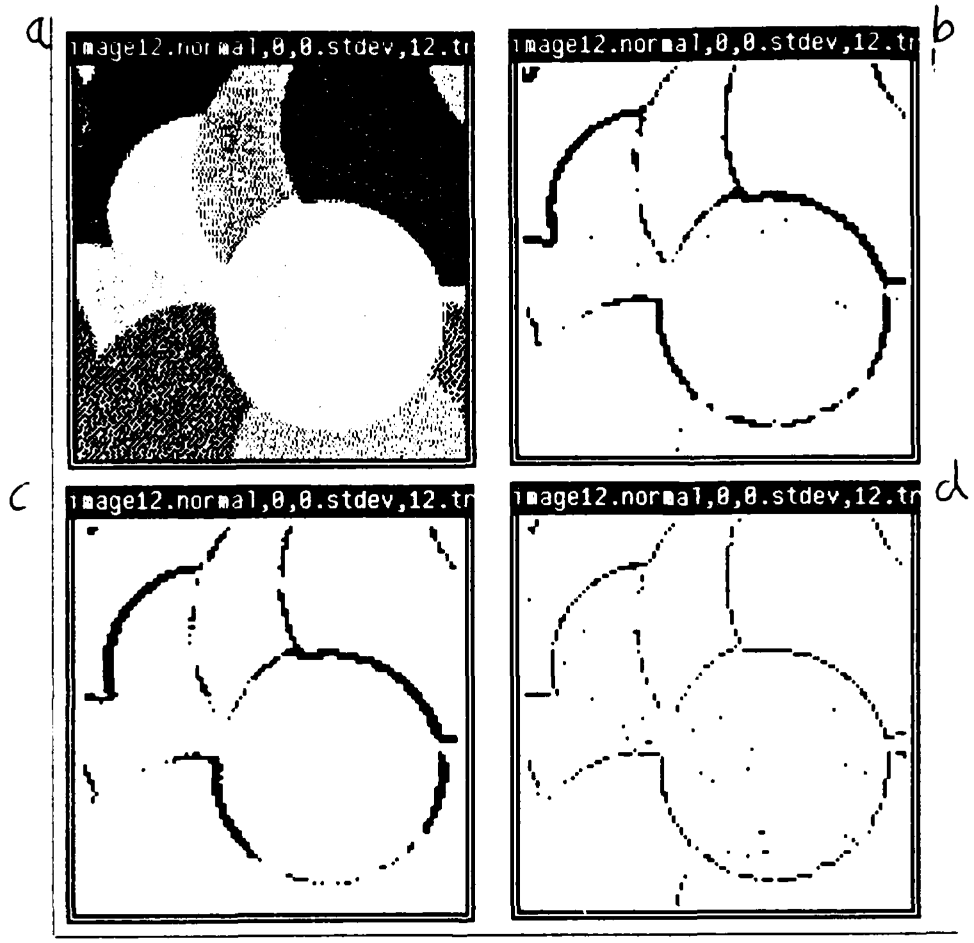
(a) Tinkertoy Image
(b) Output of the Sobel Optimally Thresholded
(c) Output of the 5x5 Kirsch Optimally Thresholded
(d) Output of the Thinned Kirsch Optimally Thresholded
Figure 34: Application of Established Operators to Tinkertoy Image


a
b
c
d
lake.window,C,100,100.iff
lake.window,C,100,100.iff
lake.window,C,100,100.iff
lake.window,C,100,100.iff

(a) Aerial Image
(b) Output of the Sobel Optimally Thresholded
(c) Output of the 5x5 Kirsch Optimally Thresholded
(d) Output of the Thinned Kirsch Optimally Thresholded
Figure 35: Application of Established Operators to Aerial Image

One can criticize these comparisons by saying that the image statistics favor my operators, which are robust with dim images. To counter this criticism I have also run the three operators (Sobel, Kirsch and thinned Kirsch) with a preprocessing stage of histogram equalization. Thus all the test images will be rescaled to have the same statistics. Thus when I find the optimal threshold for the standard deviation 12 artificial image it should remain a good threshold for all the tests.

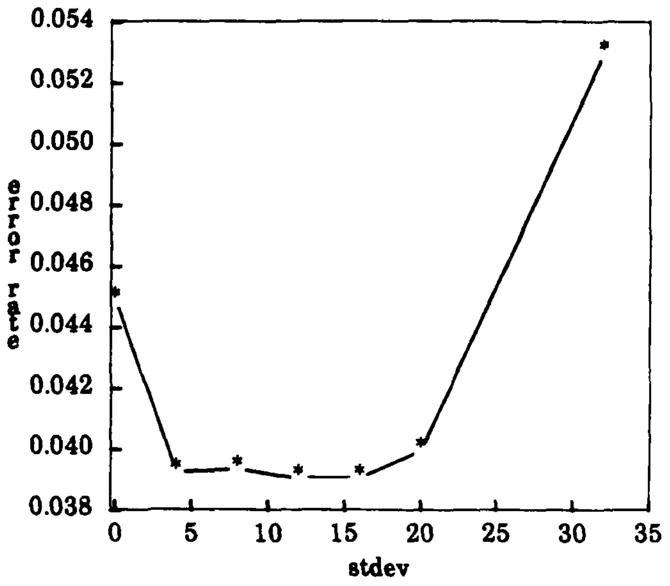
The optimal thresholds happened to be 220 for the Sobel, 750 for the Kirsch (coincidently), and 125 for the thinned Kirsch. The result of using these operators and thresholds on the standard deviation 12 artificial image (figure 18b) is shown in figure 36.


a
image12.normal,0,0.stdev,12.tr
b
image12.normal,0,0.stdev,12.tr
c
image12.normal,0,0.stdev,12.tr
d
image12.normal,0,0.stdev,12.tr

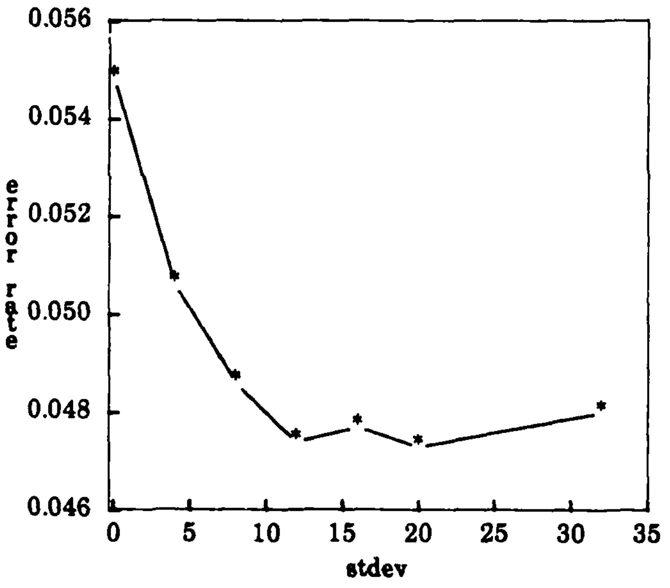
(a) Image with $\sigma=12$ noise.
(b) Output of the Histogram Equalized Sobel Optimally Thresholded
(c) Output of the Histogram Equalized 5x5 Kirsch Optimally Thresholded
(d) Output of the Histogram Equalized Thinned Kirsch Optimally Thresholded
Figure 36: Application of Histogramed Equalized Operators to Artificial Image

In figure 37 are charts that describe the result of applying these operators to the histogram equalized artificial image (figure 18b).

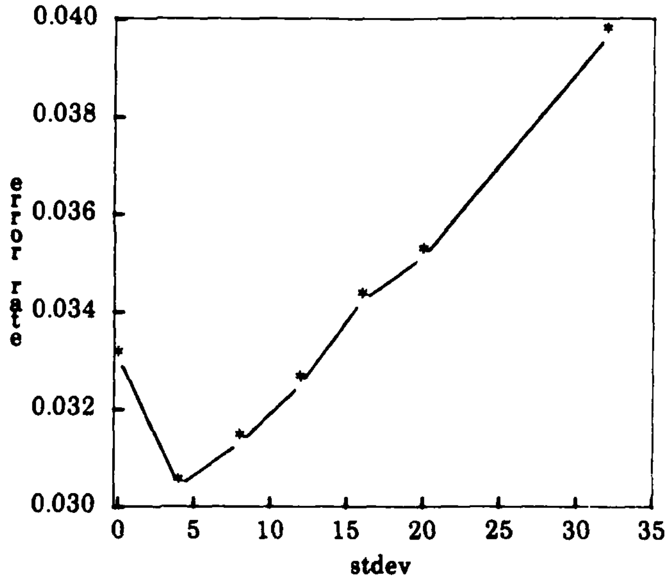
(a)


0.054
0.052
0.050
0.048
0.046
0.044
0.042
0.040
0.038
error rate
0
5
10
15
20
25
30
35
stdev

(b)

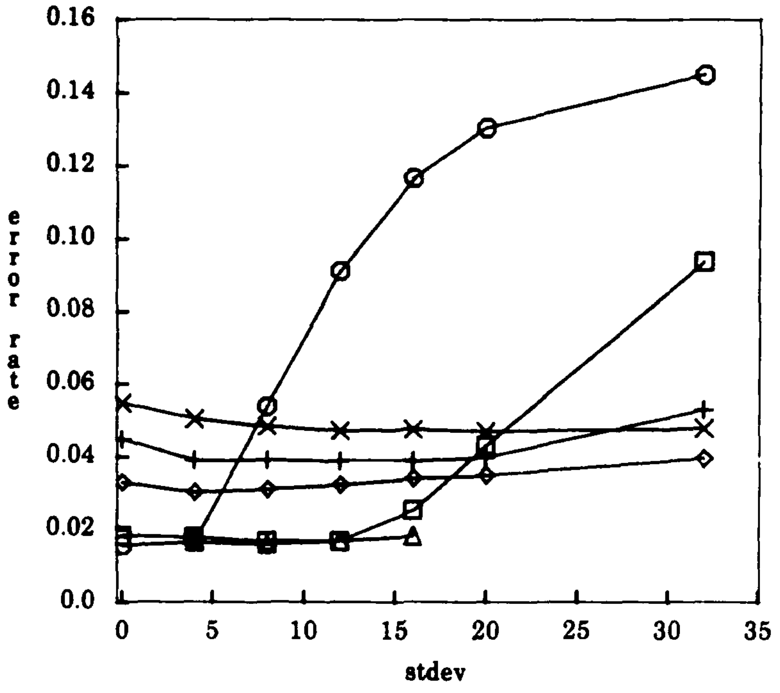

0.056
0.054
0.052
0.050
0.048
0.046
error rate
0
5
10
15
20
25
30
35
stdev

(c)



(a) Total Error rates for the thresholded Sobel on the Histogram Equalized Image
(b) Total Error rates for the thresholded Kirsch on the Histogram Equalized Image
(c) Total Error rates for the thinned Kirsch on the Histogram Equalized Image
Figure 37: Charts of Error Rates for Established Operators

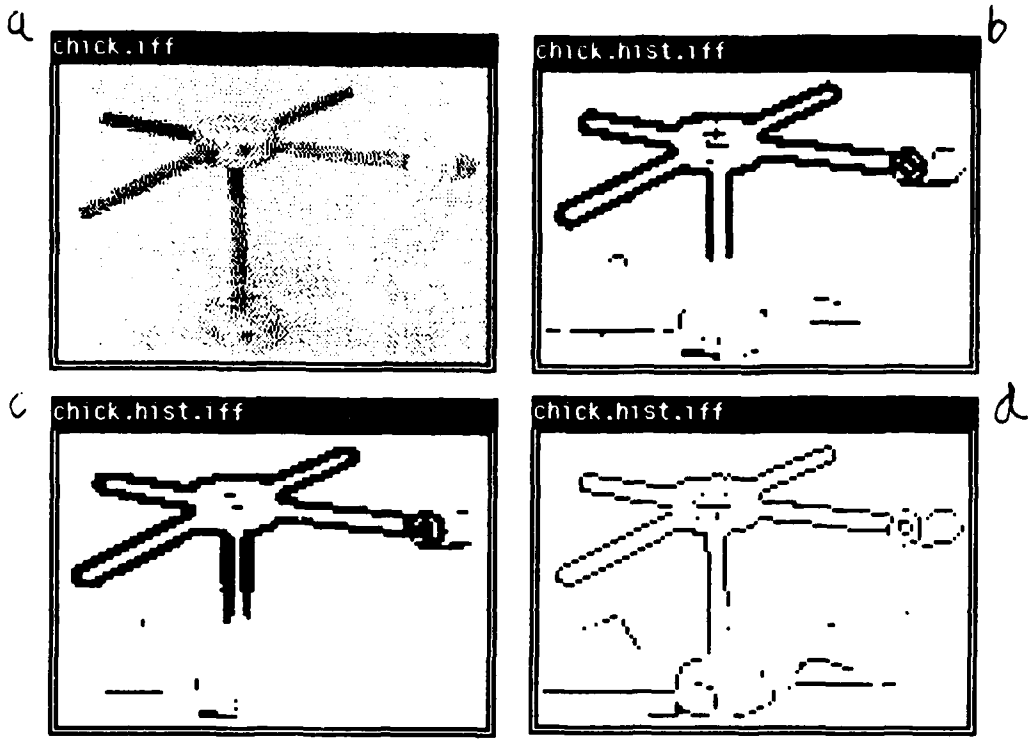
In figure 38 I compare the 3 operators on histogram equalized images to my operators on the original images (my operators do not expect histogram equalization and doing so may confuse them).



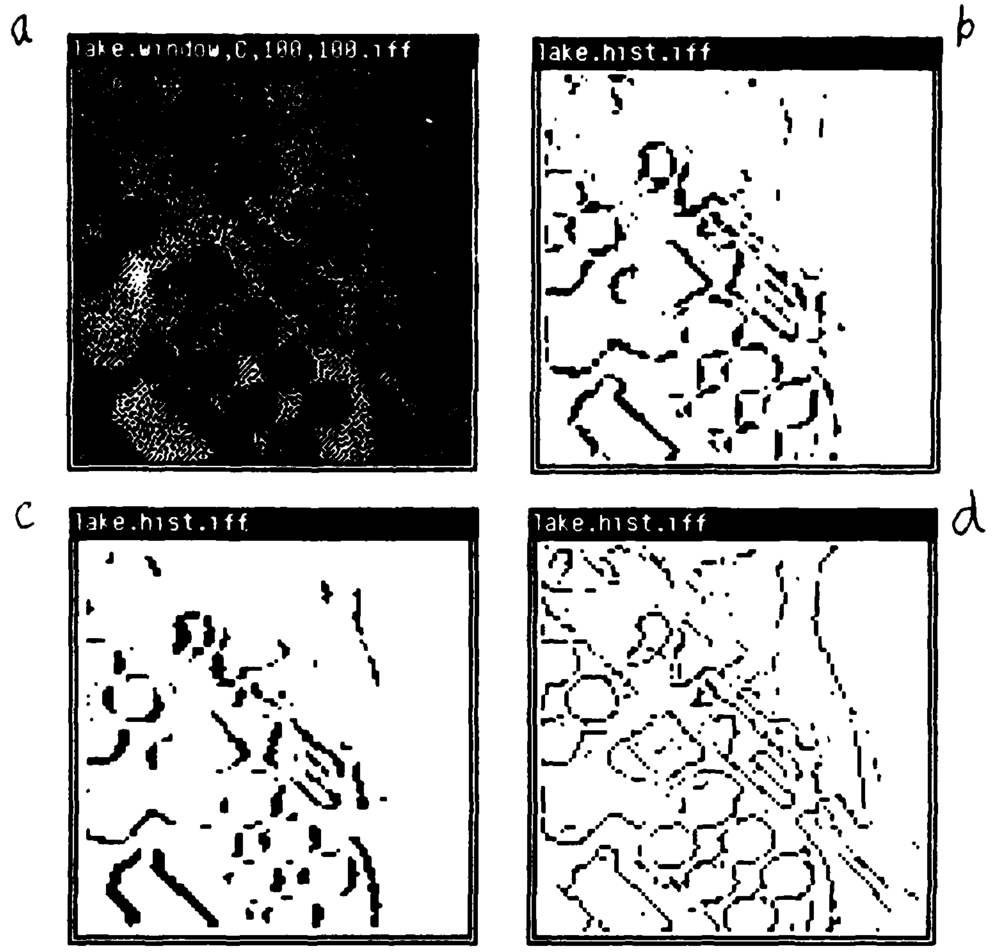
circles:	Tuned $\sigma=12$ operator
squares:	Tuned $\sigma=4$ operator
triangles:	Tuned to noise σ operator
crosses:	Histogram Equalized Sobel's error rate
X's:	Histogram Equalized 5x5 Kirsch's error rate
diamonds:	Histogram Equalized Thinned Kirsch's error rate

Figure 38: Comparison with Established Operators Applied to Histogram Equalized Image

Since the artificial images already had the full range of graylevels histogram equalization did not help the established operators much. However the advantage of histogram equalization is shown clearly when the operators are applied to real images in figures 39, and 40.



(a) Tinkertoy Image
(b) Output of the Histogram Equalized Sobel Optimally Thresholded
(c) Output of the Histogram Equalized 5x5 Kirsch Optimally Thresholded
(d) Output of the Histogram Equalized Thinned Kirsch Optimally Thresholded
Figure 39: Application of Established Operators to Tinkertoy Image



(a) Aerial Image
(b) Output of the Histogram Equalized Sobel Optimally Thresholded
(c) Output of the Histogram Equalized 5x5 Kirsch Optimally Thresholded
(d) Output of the Histogram Equalized Thinned Kirsch Optimally Thresholded
Figure 40: Application of Established Operators to Aerial Image

9. Previous Work

Edge detection is the established vision task that bears most closely on the boundary detection problem I describe here. Edge detection has been one of the earliest and most important tasks attempted by computer vision systems. Usually edge detection is described as a problem in image reconstruction.

Edge detection is often characterized as discovering the contrast in a region of the ideal image when there is an boundary between two constant intensity regions of the ideal image. Since I am not motivated by image reconstruction this task is not of particular interest for me. However often edge detection algorithms are used for boundary point detection. The idea is to accept as boundaries the pixels whose windows the detector considers to have high contrast.

The first work on edge detection was by Roberts who developed the Roberts edge operator to detect boundaries and corners of blocks. It was a simple convolution operator probably inspired by convolution based pattern matching. Since Roberts edge detection has been worked on by a large number of vision workers. Some of the operators were worked out in a somewhat ad hoc manner as the Roberts was. The "best" and most common example of such operators is the Sobel edge operator

Many have worked on "optimal operators" where some model of edges is presented and the "best" function that fits a specified functional form. The definition of "best" and the functional form varies. Almost everyone who takes this approach limits the functional forms of edge operators to convolutions.

Hueckel [Hueckel71] considers convolutions over a disc on the image. He models edges as step edges with linear boundaries occurring at random places in the disk. His functions are limited to look at only certain specific Bessel coordinates (of an integral Fourier transform) that he has determined are useful for edge detection. He takes into account somewhat the possibility of two edges in the region. In a later paper [Hueckel73] Hueckel considers edges that are two parallel step edges a few pixels apart. He analyzes such edges the same way he analyzed the previous kind of edges.

At MIT starting with Marr [Marr82] there has been concentration on zero crossing based edge detection. The edge detectors they use are to locate edges at zero crossings of a Laplacian of a Gaussian. [Torre86] [Lunscher86b] [Lunscher86a] describe how such an edge detector is an approximation to a spline based operator that has maximal output at edges (compared to elsewhere). Such an operator also has been shown always to create connected boundaries.

Canny [Canny83] has examined the issue of convolution based edge detection more closely. In particular he studied the goals of edge detection. He considered an edge detector to be good if it reported strongly when there was an edge there and *did not report* when there was no edge. He also wanted a detector that only reported once for an edge. He found that these constraints conflict when one is limited to convolution based edge detectors (such behavior arises naturally for the boundary point detectors in this paper). His primary work on this topic was with a 1 dimensional step edge model. He derived a convolution operator that was similar to a 0 crossing operator. He also discusses how to extend the operators defined for one dimensional images to two dimensional images, and when oriented operators are desirable. His operators, applied to real images, usually appear to do a good job of finding the boundaries. In this paper I derive boundary point detectors for step edges but do not constrain the functional form of the edge detector. Thus the edge detectors based on this should have performance at least as good as Canny's detector.

Nalwa [Nalwa84] used a more sophisticated model where he assumed that regions in the intensity image fit (at least locally) surfaces that are planar, cubic or tanh type. He tested whether a surface fit a window on the image and if not he tried to fit various boundaries between surfaces. He ordered the tests to be of increasing computational complexity. His operators, applied to real images, usually appear to do a good job of finding the boundaries. The work in this paper handles models of this form and derives optimal operators.

Another approach to edge detection is to simulate parts of the human early visual system. Zero crossing operators were originally motivated by this argument since it was found that there were cells in the human early visual system that compute zero crossings at various frequencies [Marr82]. Other work that seriously studies the human early visual system was by Fleet [Fleet84] on the spatio-temporal properties of center-surrounds operators in the early human visual system. My work is not concerned with the structure of the early human visual system since its goals are to perform a task best as possible rather than as human-like as possible. However I can draw inspiration from the human visual system since it has been highly optimized to its goals by evolution and hence an optimal detection system may be similar to that of the human eye (or animal eye for that matter).

Haralick has taken a similar approach to mine for the problem of edge detection [Haralick86a]. The differences between his approach and mine are that he models the image as a surface rather than as a function of a scene, and his operators generate decisions about edges rather than probabilities of edges [Haralick84]. However he has told me that his theory can be used to generate likelihoods that can be used with the techniques presented here [Haralick86b]. The relationship between his facet model and my template based models is currently under investigation.

10. Conclusion

I have demonstrated an operator that fulfills the desiderata in section 1. It has flexible output that can be used by many operators because it returns probabilities. It works on gray scale input. Because the operator is based on windows it does work proportional to the size of the image to calculate boundary probabilities. By constructing templates to represent a boundary shifted less than a pixel one can have subpixel precision with work proportional to that precision. A parameter of the algorithms I describe is the expected distribution of illuminances. Another is the standard deviation of and correlation in the noise.

Results were reported from using a 5 by 5 operator developed from this theory in section 8. I have applied this detector to artificial and real images. In section 8.3 I have compared my detectors to the established detectors, Sobel, Kirsch, and thinned Kirsch. In the next few weeks results from using a 7 by 7 and 9 by 9 operator will be available. Also comparisons will be done between these detectors and more advanced edge detectors such as Canny's [Canny83], and Haralick's [Haralick84].

In a companion report I describe an evidence combination theory that is applied to operators that return likelihoods that allows me to combine robustly the output of several different operators on the same data [Sher87]. Soon there will be results from using the likelihoods as input to a Markov random field based system [Chou87].

[Hueckel73]
M. H. Hueckel, A local Visual Operator Which Recognizes Edges and Lines, *J. ACM 20*,4 (October 1973), 634-647, ACM.

[Lunscher86a]
W. H. H. J. Lunscher and M. P. Beddoes, Optimal Edge Detector Design II: Coefficient Quantization, *Pattern Analysis and Machine Intelligence 8*,2 (March 1986), 178-187, IEEE.

[Lunscher86b]
W. H. H. J. Lunscher and M. P. Beddoes, Optimal Edge Detector Design I: Parameter Selection and Noise Effects, *Pattern Analysis and Machine Intelligence 8*,2 (March 1986), 164-177, IEEE.

[Marr82] D. Marr, *Vision*, W. H. Freeman and Company., New York, 1982

[Marroquin85]
J. L. Marroquin, Probabilistic Solution of Inverse Problems, Tech. Rep. 860, MIT Artificial Intelligence Laboratory, September 1985.

[Nalwa84] V. S. Nalwa, On Detecting Edges, *Proceedings: Image Understanding Workshop*, October 1984, 157-164.

[Sher85] D. B. Sher, Evidence Combination for Vision using Likelihood Generators, *Proceedings: Image Understanding Workshop (DARPA)*, Miami, Florida, December 1985, 255-270. Sponsored by: Information Processing Techniques Office Defence Advanced Research Projects Agency.

[Sher86] D. Sher, Optimal Likelihood Detectors for Boundary Detection Under Gaussian Additive Noise, *IEEE Conference on Computer Vision and Pattern Recognition*, Miami, Florida, June 1986.

[Sher87] D. B. Sher, Evidence Combination Based on Likelihood Generators, TR192, University of Rochester Computer Science Department, Milan, Italy, January 1987. Submitted in shorter form to IJCAI.

[Torre86] V. Torre and T. A. Poggio, On Edge Detection, *Pattern Analysis and MAchine Intelligence* 8,2 (March 1986), 147-163, IEEE.

END

6-87

DTIC